University of Alberta

Support Vector Machines, Radial Basis Functions Neural Networks, and Fuzzy Granulation in Pattern Recognition

by

Lino Martin Ramirez

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta

Spring, 2002

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-69750-9

Canada

University of Alberta

Library Release Form

Name of Author: Lino Martin Ramirez

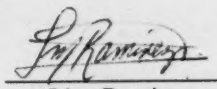
Title of Thesis: Support Vector Machines, Radial Basis Functions Neural Networks, and Fuzzy Granulation in Pattern Recognition

Degree: Master of Science

Year this Degree Granted: 2002

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



Lino Ramirez
214E Michener Park NW
Edmonton, Alberta
Canada, T6H 4M5

Date: December 13, 2001

…The search for truth, even when it concerns
a finite reality of the world or of man,
is never-ending, but always points beyond
to something higher than the immediate
object of study, to the questions which give
access to mystery…

John Paul II

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled "**Support Vector Machines, Radial Basis Functions Neural Networks, and Fuzzy Granulation in Pattern Recognition**" submitted by **Lino Martin Ramirez** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dr. W. Pedrycz (Supervisor)

Dr. A. Robinson (External)

Dr. M. Reformat (Internal)

Date: December 13, 2001

Dedicated to my wife, Marlene

Abstract

Pattern recognition is a central task in a variety of applications ranging from engineering to financial analysis. For this reason, it is very important to develop efficient pattern recognition systems that help to make decisions automatically and reliably. This thesis describes the implementation of pattern recognition systems based on computational intelligence approaches (in particular support vector machines and radial basis function neural networks). To improve the efficacy of these systems, a notion of prototypes stability analysis is introduced and fuzzy kernels are developed. Prototypes stability analysis is used for determining the adequate number of prototypes, clusters, or information granules to be used in classifiers design. Fuzzy kernels, optimized using genetic algorithms, allow for the incorporation of fuzzy set methods into the support vector machine approach. To demonstrate the applicability of these pattern recognition systems, detailed numerical experiments with synthetic and real-world data sets are included and analyzed.

Acknowledgement

I would like to thank my supervisor Dr. Witold Pedrycz for his guidance, support, and encouragement throughout these years of graduate study.

I would like to express my gratitude to Dr. Nicolino Pizzi for his support in the development of an important part of this research.

All my love and thanks go to my wife, Marlene, for her love, support, and encouragement.

Finally, I would like to thank my parents Zoraida and Lino, and my parents in-law Marlene and Abbes for their constant support.

Table of Content

1 Introduction **1**

1.1 Pattern recognition *1*

1.2 Thesis overview *2*

2 Problem Overview **4**

2.1 Introduction *4*

2.2 Pattern recognition *4*

2.3 The pattern recognition system *4*

2.4 Performance measures *6*

2.5 Approaches for pattern recognition *7*

2.5.1 Template matching 7

2.5.2 Statistical methods 7

2.5.3 Structural or syntactic methods 7

2.5.4 Computational intelligence 8

2.6 Data sets to be studied *8*

2.6.1 Synthetic data sets 8

2.6.2 Iris data set 10

2.6.3 Wisconsin breast cancer database 11

2.6.4 Boston housing data set 12

2.6.5 A Severe storm cell data set 13

2.7 Summary *14*

3 Granular Computing **15**

3.1 Introduction *15*

3.2 Information granulation *15*

3.3 Construction of information granules *17*

3.4 Prototypes stability analysis to determine an appropriate number of information granules to be used in classifiers *20*

3.5 Illustrative examples *26*

3.5.1 Synthetic data sets 26

3.5.2 Iris data set 29

3.6 Summary *29*

4 Support vector machines (SVM's) in pattern recognition 30

4.1 Introduction *30*

4.2 Theory *30*

4.2.1 The maximal margin classifier 31

4.2.2 The linear soft-margin classifier 34

4.2.3 The non-linear classifier 35

4.3 Illustrative examples *37*

4.4 Model selection *38*

4.5 Multi-class SVM *45*

4.5.1 1-v-R 45

4.5.2 DDAG 45

4.6 Implementation techniques *47*

4.6.1 General Issues 47

4.6.2 The Gradient ascent approach 48

4.6.3 Chunking and Decomposition 49

4.6.4 Sequential minimal optimization 49

4.7 Relationship of SVM to partitional clustering *50*

4.8 Summary *50*

5 Radial basis functions networks in pattern recognition 51

5.1 Introduction *51*

5.2 Background *51*

5.3 Classic radial basis function (CRBF) neural networks 52

5.4 CRBF neural network training 53

5.5 An empirical comparison of the SVM with the RBF against CRBF 54

5.6 Summary 61

6 Designing Support Vector Machines with the Use of Fuzzy Granulation: Some Preliminary Results **62**

6.1 Introduction 62

6.2 Fuzzy sets and support vector machines 62

6.3 Generation of information granules 63

6.4 Fuzzy kernels 64

6.4.1 Derivation 64

6.4.2 Fuzzy kernel outputs 67

6.4.3 Fuzzy kernels optimization 68

6.4.3.1 Background 68

6.4.3.2 Optimization algorithm 70

6.5 General Scheme 73

6.6 Numerical studies 75

6.7 Summary 80

7 Experimental Studies **81**

7.1 Introduction 81

7.2 Architectures 81

7.3 Iris Classification 83

7.3.1 Methodology 84

7.3.2 Comparative analysis 84

7.4 Wisconsin Breast Cancer Classification 86

7.4.1 Methodology 87

7.4.2 Comparative analysis 87

7.5 Boston Housing Classification *89*
7.5.1 Methodology 89
7.5.2 Comparative analysis 90
7.6 Storm Cell Classification *92*
7.6.1 Weather data (a): Hail vs. Tornado 92
7.6.1.1 Methodology 92
7.6.1.2 Comparative analysis 93
7.6.1.3 Comparison with previous work 95
7.6.2 Weather data (b): Four-class Classification Problem 96
7.6.2.1 Methodology 96
7.6.2.2 Comparative analysis 96
7.6.3 Weather data (c): Ten-class Classification Problem 99
7.6.3.1 Methodology 99
7.6.3.2 Comparative analysis 100
7.7 Discussion *109*
7.8 Summary *109*
8 Conclusions and Recommendations for Future Work **111**
8.1 Conclusions *111*
8.2 Future Work *112*
References **114**

List of Tables

Table 2-1: Examples of Pattern Recognition Applications 5
Table 2-2: Confusion matrix for hypothetical results in a test example. 6
Table 2-3: Listing of features for the iris data set. 10
Table 2-4: Listing of features for the Wisconsin breast cancer database 12
Table 2-5: Listing of features for the Boston housing data set. 13
Table 2-6: Listing of derived features for the severe storm cell data set. 14
Table 3-1: Results obtained by clustering the n_p portions of the data. 23
Table 3-2: Algorithm for applying PSA. 25
Table 3-3: Number of clusters (*nc*) to be used in clustering data (a). 27
Table 3-4: Number of clusters (*nc*) to be used in clustering data (b). 28
Table 3-5: Number of clusters (*nc*) to be used in clustering data (c). 28
Table 3-6: Number of clusters (*nc*) to be used in clustering the iris data. 29
Table 4-1: Some commonly used kernels 35
Table 4-2: Algorithm for setting the parameters of the radial basis function kernel. 44
Table 5-1: Algorithm for setting the parameters of the radial basis function kernel. 56
Table 5-2: Set of parameters to be used in the process of cross-validation, in the case of the SVM, for the artificial-data classification problem. 57
Table 5-3: Algorithm for setting the spread of a radial basis function neural network. 57
Table 5-4: Set of parameters to be used in the process of cross-validation, in the case of the CRBF1, for the artificial-data classification problem. 58
Table 5-5: Set of parameters to be used in the process of cross-validation, in the case of the CRBF2, for the artificial-data classification problem. 58
Table 5-6: Classification results obtained with different classifiers for data (a). 58
Table 5-7: Classification results obtained with different kernels for data (b). 59
Table 6-1: Rule bases for the control of p_c. 72
Table 6-2: Rule bases for the control of p_m. 73
Table 6-3: Algorithm for implementing the HSVM approach. 75
Table 6-4: Classification results obtained with the HSVM. 80
Table 7-1: Set of parameters to be used in the process of cross-validation, in the case of the SVM, for the real-data classification problems. 82
Table 7-2: Set of parameters to be used in the process of cross-validation, in the case of the CRBF1, for the real-data classification problems. 83
Table 7-3: Set of parameters to be used in the process of cross-validation, in the case of the CRBF2, for the real-data classification problems. 83
Table 7-4: CRBF1 training and testing performance results for the iris data set. 84
Table 7-5: CRBF2 training and testing performance results for the iris data set. 85
Table 7-6: 1-NN training and testing performance results for the iris data set. 85
Table 7-7: DDAG training and testing performance results for the iris data set. 85
Table 7-8: 1-v-R training and testing performance results for the iris data set. 85
Table 7-9: Summary of results for the iris data set. 86
Table 7-10: CRBF1 training and testing performance results for the Wisconsin breast cancer data. 87
Table 7-11: CRBF2 training and testing performance results for the Wisconsin breast cancer data. 88

Table 7-12: 1-NN training and testing performance results for the Wisconsin breast cancer data. 88
Table 7-13: SVM training and testing performance results for the Wisconsin breast cancer data. 88
Table 7-14: Summary of results for the Wisconsin breast cancer data set. 88
Table 7-15: CRBF1 training and testing performance results for the Boston housing data. ... 90
Table 7-16: CRBF2 training and testing performance results for the Boston housing data. ... 90
Table 7-17: 1-NN training and testing performance results for the Boston housing data. 91
Table 7-18: SVM training and testing performance results for the Boston housing data. 91
Table 7-19: Summary of results for the Boston housing data set. 91
Table 7-20: CRBF1 training and testing performance results for the weather data (a). 93
Table 7-21: CRBF2 training and testing performance results for the weather data (a). 93
Table 7-22: 1-NN training and testing performance results for the weather data (a). 94
Table 7-23: SVM training and testing performance results for the weather data (a). 94
Table 7-24: Summary of results for the weather data set (a). 94
Table 7-25: Performance of different methods for solving a storm cell classification problem. 95
Table 7-26: CRBF1 training and testing set performance results for the weather data (b). 97
Table 7-27: CRBF2 training and testing set performance results for the weather data (b). 97
Table 7-28: 1-NN training and testing set performance results for the weather data (b). 97
Table 7-29: DDAG training and testing set performance results for the weather data (b). 97
Table 7-30: 1-v-R training and testing set performance results for the weather data (b). 98
Table 7-31: Summary of results for the weather data (b). 99
Table 7-32: Pattern distribution for the 10-class classification problem 100
Table 7-33: CRBF1 training set performance results (average) for the weather data (c). 101
Table 7-34: CRBF1 training set performance results (standard deviation) for the weather data (c). 101
Table 7-35: CRBF1 testing set performance results (average) for the weather data (c). 101
Table 7-36: CRBF1 testing set performance results (standard deviation) for the weather data (c). 102
Table 7-37: CRBF2 training set performance results (average) for the weather data (c). 102
Table 7-38: CRBF2 training set performance results (standard deviation) for the weather data (c). 102
Table 7-39: CRBF2 testing set performance results (average) for the weather data (c). 103
Table 7-40: CRBF2 testing set performance results (standard deviation) for the weather data (c). 103
Table 7-41: 1-NN training set performance results (average) for the weather data (c). 103
Table 7-42: 1-NN training set performance results (standard deviation) for the weather data (c). 104
Table 7-43: 1-NN testing set performance results (average) for the weather data (c). 104
Table 7-44: 1-NN testing set performance results (standard deviation) for the weather data (c). 104
Table 7-45: DDAG training set performance results (average) for the weather data (c). 105
Table 7-46: DDAG training set performance results (standard deviation) for the weather data (c). 105
Table 7-47: DDAG testing set performance results (average) for the weather data (c). 105
Table 7-48: DDAG testing set performance results (standard deviation) for the weather data (c). 106
Table 7-49: 1-v-R training set performance results (average) for the weather data (c). 106

Table 7-50: 1-v-R training set performance results (standard deviation) for the weather data (c). 106
Table 7-51: 1-v-R testing set performance results (average) for the weather data (c)..... 107
Table 7-52: 1-v-R testing set performance results (standard deviation) for the weather data (c). 107
Table 7-53: Summary of results for the weather data (c). 108

List of Figures

Figure 2-1: A typical scheme of pattern recognition. 5
Figure 2-2: Synthetic data (a). Squares and diamonds represent class 1 and class 2 respectively. 9
Figure 2-3: Synthetic data (b). Squares and diamonds represent class 1 and class 2 respectively. 9
Figure 2-4: Synthetic data (c). Five clusters with unit covariance matrices. 10
Figure 2-5: A two-dimensional (dimension 1 vs. dimension 2) section of the Anderson iris data 11
Figure 2-6: A two-dimensional (dimension 3 vs. dimension 4) section of the Anderson iris data 11
Figure 3-1: A granule is a set of objects that are grouped based on their similarity, proximity or functionality. 15
Figure 3-2: Examples of crisp and fuzzy granulation 16
Figure 3-3: Basic structure of fuzzy information granulation: granules, attributes, and values. 17
Figure 3-4: Synthetic data (c). 21
Figure 3-5: Prototypes generated by the FCM with c=5 21
Figure 3-6: Prototypes generated by the FCM with c=4 22
Figure 3-7: Two overlapping clusters. 26
Figure 3-8: Synthetic data (a). Squares and diamonds represent class 1 and class 2 respectively. 27
Figure 3-9: Synthetic data (b). Squares and diamonds represent class 1 and class 2 respectively. 27
Figure 3-10: Synthetic data (c). 28
Figure 4-1: Illustrative classification problem. 31
Figure 4-2: An optimal margin classifier. The dashed lines identify the margin $M=1/\|\mathbf{w}\|$. The patterns inside the squares are the support vectors. 32
Figure 4-3: A soft margin classifier. The dashed lines identify the margin. The patterns inside the square are the support vectors. ξ_1 and ξ_{12} are the positive variables associated with patterns 1 and 12 respectively. All the other ξ_i are equal to zero. 34
Figure 4-4: A non-linear classifier. The mapping from the input space (top left) to the feature space (top middle) is performed using the function Φ. The separating hyperplane in the feature space (top middle), as determined by the SVM (bottom), corresponds to a non-linear decision surface in the input space (top right). 36
Figure 4-5: Synthetic data. Squares and diamonds represent class 1 and class 2 respectively. 37
Figure 4-6: The discrimination boundary produced by the SVM with the linear kernel for C=100. The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern 37
Figure 4-7: The discrimination boundaries produced by the SVM with the polynomial kernel (d=2) for C=100. The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern. 38
Figure 4-8: The discrimination boundary produced by the SVM with the radial basis function kernel (γ=0.04) for C=100. The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern. 38

Figure 4-9: Decision boundaries generated by the SVM with the linear kernel for C=0.1 (a), C=1 (b), C=10 (c), and C=1000 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern. 39
Figure 4-10: Decision boundaries generated by the SVM with the polynomial kernel (C=100) for d=2 (a), d=3 (b), d=5 (c), and d=10 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern. 40
Figure 4-11: Decision boundaries generated by the SVM with the polynomial kernel (d=2) for C=0.1 (a), C=1 (b), C=10 (c), and C=1000 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern. 41
Figure 4-12: Decision boundaries generated by the SVM with the RBF kernel (C=100) for γ=0.005 (a), γ=0.05 (b), γ=0.5 (c), and γ=5 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern. 42
Figure 4-13: Decision boundaries generated by the SVM with the RBF kernel (γ=0.1) for C=0.1 (a), C=1 (b), C=10 (c), and C=1000 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern. 43
Figure 4-14: DDAG 46
Figure 5-1: Feed-forward (a) and Feedback (b) neural network architectures. 52
Figure 5-2: Architecture of a radial basis function neural network. $\mathbf{x}_i$ is the ith p-dimensional input pattern. Ns is the number of centres. k is the number of classes. 52
Figure 5-3: Schematic example of data points that fall into four distinct classes. 53
Figure 5-4: Synthetic data (a). Squares and diamonds represent class 1 and class 2, respectively. 55
Figure 5-5: Synthetic data (b). Squares and diamonds represent class 1 and class 2, respectively. 55
Figure 5-6: Decision boundaries for the SVM on data (a). Testing points are the bold patterns. Support vectors are indicated with a circle around the corresponding patterns. 59
Figure 5-7: Decision boundaries for the CRBF1 on data (a). Testing points are the bold patterns. The centre is indicated with a circle around the corresponding pattern. 60
Figure 5-8: Decision boundaries for the CRBF2 on data (a). Testing points are the bold patterns. Centres are indicated with a circle. 60
Figure 6-1: The scheme of pattern recognition using hybrid support vector machines (HSVM). 63
Figure 6-2: Fuzzy kernel for fuzzy inputs. The aggregation is disjunctive, m_1 and m_2 are the outputs of the matching neurons, and w_1 and w_2 are the aggregative neuron's connections. 64
Figure 6-3: Fuzzy kernel for fuzzy inputs. The aggregation is conjunctive, m_1 and m_2 are the outputs of the matching neurons, and w_1 and w_2 are the aggregative neuron's connections. 65
Figure 6-4: Fuzzy kernel as a superposition of matching and aggregative computation. In this case, the aggregation is disjunctive. 66
Figure 6-5: Mapping from a $\Re^p$ space of the input data to the [0,1] space of the kernel output, where p is the dimensionality of the input data and c is the number of information granules. 67
Figure 6-6: Non-linear characteristic of the kernel with conjunctive aggregation, with w_1=0.3, w_2=0.3, b_1=0.4, b_2=0.5, $a_1 \in [0,1]$, $a_2 \in [0,1]$, and the algebraic product and probabilistic sum norms. 67

Figure 6-7: Non-linear characteristic of the kernel with disjunctive aggregation, with w_1=0.3, w_2=0.3, b_1=0.4, b_2=0.5, $a_1 \in [0,1]$, $a_2 \in [0,1]$, and the algebraic product and probabilistic sum norms. 68
Figure 6-8: Major steps in a genetic algorithm. 70
Figure 6-9: Fuzzy kernel parameters encoding. C is the error penalty term, w_{ij} is the weight of the connection ij, c is the number of information granules, p is the number of dimensions of the input data. 71
Figure 6-10: Hybrid Support Vector Machine. 74
Figure 6-11: The hybrid support vector machine has two layers. In the first layer, the input data is mapped to the fuzzy space (if required). During the learning process, the first layer selects the basis $K(x_j, x_i)$ (for i=1,…,n and j=1,…,nsv) as well as the number nsv, which corresponds to the number of support vectors. The second layer performs a linear combination of the outputs of the first layer. 74
Figure 6-12: Synthetic data (a). Squares and diamonds represent class 1 and class 2 respectively. 76
Figure 6-13: Synthetic data (b). Squares and diamonds represent class 1 and class 2 respectively. 77
Figure 6-14: The discrimination boundaries produced by the HSVM classifier for data (a). Testing points are the bold patterns. Support vectors are indicated with a circle around the corresponding patterns. 78
Figure 6-15: Fitness in successive generations for the data displayed in Figure 6-13. 78
Figure 6-16: The discrimination boundaries produced by the HSVM classifier for data (b). Testing points are the bold patterns. Support vectors are indicated with a circle around the corresponding training patterns. Errors in testing are indicated with an X across the testing pattern with the wrong shape. 79
Figure 6-17:Fitness in successive generations for the data displayed in Figure 6-15. 79
Figure 7-1: Training accuracy for the iris data set. 86
Figure 7-2: Testing accuracy for the iris data set. 86
Figure 7-3: Training accuracy for the Wisconsin breast cancer data set. 89
Figure 7-4: Testing accuracy for the Wisconsin breast cancer data set. 89
Figure 7-5: Training accuracy for the Boston housing data set. 91
Figure 7-6: Testing accuracy for the Boston housing data set. 92
Figure 7-7: Training accuracy for the weather data (a). The classification problem is binary. 94
Figure 7-8: Testing accuracy for the weather data (a). The classification problem is binary. 95
Figure 7-9: Training accuracy for the weather data (b). 4-class case. 98
Figure 7-10: Testing accuracy for the weather data (b). 4-class case. 99
Figure 7-11: Training accuracy for the weather data (c). 10-class case. 108
Figure 7-12: Testing accuracy for the weather data (c). 10-class case. 108

List of Abbreviations and Symbols

1-NN	1-nearest neighbour
1-v-R	One versus the Rest
CI	Computational Intelligence
CM	Confusion Matrix
CRBF	Classic Radial Basis Function
CRBF1	Classic Radial Basis Function neural network with the centres found using the OLS approach
CRBF2	Classic Radial Basis Function neural network with the centres found using FCM
Crisp IG	Crisp Information Granulation
DDAG	Decision-Directed Acyclic Graph
E	Error
FCM	Fuzzy c-means
Fuzzy IG	Fuzzy Information Granulation
GA	Genetic Algorithm
HSVM	Hybrid Support Vector Machine
IG	Information Granulation
K	Kernel
k-NN	k-nearest neighbours
m	Fuzzification factor for the FCM
MLP	Multilayer perceptron
n	Number of elements
nl	Number of linguistic terms
nsv	Number of Support Vectors
OLS	Orthogonal Least Squares
OSH	Optimal Separating Hyperplane
p	Number of dimensions
p_c	Probability of Crossover
PCA	Principal Component Analysis
p_m	Probability of mutation
P_o	Classification accuracy
PS	Population size
PSA	Prototypes Stability Analysis
RBF	Radial Basis Function
SOM	Self Organizing Maps
SV	Support Vector
SVM	Support Vector Machine
SVM's	Support Vector Machines
κ	Kappa score

1 Introduction

1.1 Pattern recognition

Pattern recognition generally refers to "assigning an object to a so far unknown class of objects, and identifying an object as a member of the already known class"[73]. This field of research can be applied to solve a broad range of problems related to weather forecasting, speech recognition, credit appraisal, face recognition, and so on [38]. Some of the best-known approaches for pattern recognition are syntactic matching, statistical classification, template matching, and computational intelligence methods. Such methods are able to perform classifications from labelled training data sets as well as to explore structures and classes in unlabeled data. This thesis studies the following techniques: granular computing, support vector machines, and neural networks.

Granular computing [78]-[80] is oriented toward the representation of knowledge related to problems in basic entities known as information granules. An information granule is a set of objects that are grouped based on their similarity, proximity or functionality. These information granules may be crisp (i.e. with sharp boundaries) or fuzzy (i.e. with boundaries that are not well-defined). The importance of granular computing lies in the fact that a suitable granulation may facilitate the identification of useful patterns in a data set.

The Support Vector Machine (SVM) [12], [71] is a binary classification tool that has been successfully applied in numerous areas ranging from bioinformatics to face identification. In the SVM approach, the optimization criterion is the maximization of the margin which is the minimum distance from any training point to the separating hyperplane. The location and slope of this hyperplane are defined by a group of training points called support vectors.

An artificial neural network [9], [41] is a computational structure that is composed of a number of simple processors connected through a set of links that have some weights associated with them. The radial basis function (RBF) network [9] is a neural network with two layers. The first layer consists of neurons equipped with basis functions, which normally are Gaussian-like functions. The second layer performs the linear combination of the outputs obtained from the first layer. The learning process in RBF networks involves the modification of the network architecture (centres) and the connection weights in order to classify correctly the given input patterns.

This thesis has two objectives.

1. The first one is to study the potential of RBF networks and SVM's when applied to a series of classification problems. The potential of these approaches (in contrast to classical pattern recognition approaches) lies in their ability to learn complex input-output relationships and to adapt themselves to the data. Moreover, they have been extensively evaluated and demonstrated to be useful in practical applications.
2. The second objective is to enhance the RBF networks and the SVM's by using two new concepts related to information granulation. The first concept is prototypes stability. Prototypes obtained by applying a clustering algorithm to different portions of the data should no differ significantly (i.e., they should be stable). This is not the case if there is an error in the choice of the number of clusters [65]. Based on this idea, prototypes stability analysis may be used to determine the appropriate number of clusters, prototypes or information granules to be used in classifiers design. The second new concept is fuzzy kernels. Fuzzy kernels may be used to incorporate fuzzy sets methods into the SVM approach.

To verify the proposed architectures and learning methods, extensive computer simulations were performed. The data sets used for the experiments include synthetic data sets, an iris data set, a Wisconsin breast cancer data set, a Boston housing data set, and a severe storm cell data set from Environment Canada.

1.2 Thesis overview

This thesis begins with an overview of the problem in Chapter 2. This chapter introduces the basic terminology needed for the upcoming chapters and the data sets to be used in the experiments (synthetic data sets, a Wisconsin breast cancer data set, an iris data set, a Boston housing data set, and a severe storm cell data set from Environment Canada).

In Chapter 3, an overview of granular computing is presented. The main interest of this chapter is to describe the use of granular computing as a tool for classifier design. In this sense, the use of prototype stability analysis for determining the best number of centres to be used in classifier design is presented.

Chapter 4 studies the Support Vector Machine classifier and its extension to the multi-class case. Some illustrative examples are discussed.

An introduction to radial basis function networks is presented in Chapter 5. Special attention is given to the methods of assigning the centres of the receptive fields. To do so, two methods are used: orthogonal least squares and a clustering approach discussed in Chapter 3. An empirical comparison of the SVM with the RBF kernel against the RBF approaches is also presented.

In Chapter 6, the original SVM approach is extended to incorporate fuzzy sets methods. The fuzzy sets are generated using fuzzy clustering. Some numerical studies involving synthetic data sets are presented.

Chapter 7 is devoted to experimental studies with a Wisconsin breast cancer data set, a Boston housing data set and a severe storm cell data set from Environment Canada to demonstrate the applicability of the developed classifiers in solving these pattern recognition problems.

The last chapter, Chapter 8, contains the conclusions resulting from this research, the summary of contributions and the recommendations for future work.

2 Problem Overview

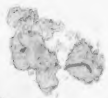
2.1 Introduction

This chapter begins with a brief introduction to pattern recognition. For further information, the reader is referred to [23], [28], [38], [9], [67]. Some performance measures are then presented, followed by a description of some of the approaches for pattern recognition. Finally, the data sets for the experiments are described.

2.2 Pattern recognition

One of the main tasks of a pattern recognition system consists of finding structure in a data set to perform classification. The classification tasks are done over a certain group of elements known as patterns. Patterns are entities characterized by a series of features. Given a pattern, the classification may be (1) supervised if the system uses the pattern-class information (i.e. the samples are labelled), or (2) unsupervised if the system does not know or does not use the pattern-class information (in this case, the patterns are assigned to a previously unknown class) [38].

Interest in the area of pattern recognition is related to its use as one of the principal tools in human decision-making tasks, helping doctors in the diagnosis of diseases, meteorologists in the forecasting of weather, bankers in the appraisal of credits, and so on (Table 2-1).



2.3 The pattern recognition system

The process of pattern recognition can be divided in three principal steps: (1) data acquisition, (2) feature selection, and (3) classification. See Figure 2-1.

1. Data acquisition. In this module, the input data are gathered via a set of sensors and converted into a form suitable for machine processing. It is important to mention that this block is highly related to the particular application that is being developed.
2. Feature selection. Usually, this block is mainly concerned with the reduction of space dimensionality. This may be stated as follows: "given a set of d features, select a subset of size m that leads to the smallest classification error" [38].

3. Classification: This module has two different modes of operation: the learning mode and the decision-making mode. In the learning mode, this module is trained to partition the feature space. This means that some parameters in this module are adjusted to produce a correct output over the training samples. In the decision-making mode, the input patterns are assigned to a specific class based on the parameters that were learned in the previous mode.

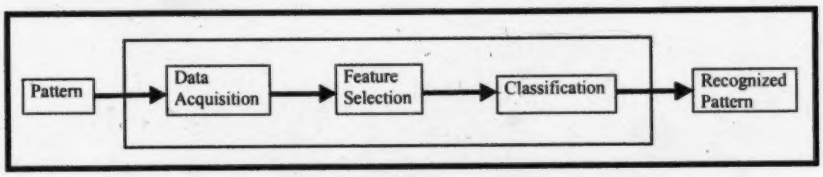


Figure 2-1: A typical scheme of pattern recognition.

Problem Domain	Application	Input Pattern	Pattern Class
Medical diagnosis	Medical waveform classification	Medical recording such as EEG and ECG	Known type of waveform
Meteorology	Severe storm cell classification	Multidimensional derived features	Known type of storm cell
Financial assessment	Credit appraisal	Financial records	Credit approval/disapproval
Multimedia retrieval	Internet search	Images	Image categories
Computer Vision	Face detection in images	Digitized image	Presence/absence of humans
Bioinformatics	Sequence analysis	DNA/Protein sequence	Known types of genes/patterns
Industrial automation	Printed circuit board inspection	Board image	Defective/non-defective nature of product
Biometric recognition	Personal identification	Face, iris, fingerprint, voice.	Authorized users for access control
Document classification	World Wide Web search	Text document	Categories of web pages, such as business, sports, entertainment, etc

Table 2-1: Examples of Pattern Recognition Applications

2.4 Performance measures

The typical classifier performance measures are classification accuracy and classification error. The classification accuracy (P_o) is equal to the number of correctly classified samples (*correct*) divided by the number of samples (n), that is

$$P_o = correct / n \tag{2-1}$$

while the classification error is

$$E = 1 - P_o \tag{2-2}$$

These performance measures may be complemented with a visual indication (of how well the classifier was trained) known as confusion matrix. The confusion matrix gives information about overall performance and distribution of errors. For example, in Table 2-2, it can be seen that there is no confusion between classes A and B, virtually no confusion between B and C, and a small confusion between A and C.

Correct Class	Classified as A	Classified as B	Classified as C
A	**117**	0	12
B	0	**113**	2
C	13	1	**106**

Table 2-2: Confusion matrix for hypothetical results in a test example.

One additional performance measure to be used in this thesis is known as kappa score (κ) [26]. The kappa score is a coefficient that is not affected by differences in the number of patterns of each class. This coefficient is given by

$$\kappa = (P_o - P_c)/(1 - P_c) \tag{2-3}$$

where (given that CM_{ii} is one element in the diagonal of the confusion matrix; CM_{ij} is an element of the confusion matrix; k is the number of classes; and n is the number of elements) P_o is the conventional performance measure

$$P_o = \left(\sum_{i=1}^{k} CM_{ii} \right) \Big/ n$$

(2-4)

and Pc is the coefficient that eliminates the agreement due to chance

$$P_c = \sum_{i=1}^{k} \left(\sum_{j=1}^{k} CM_{ij} \sum_{j=1}^{k} CM_{ji} \right) \Big/ n^2$$

(2-5)

A final note regarding the kappa score is that the closer the value is to one "1", the better the classifier is.

2.5 Approaches for pattern recognition

Some of the most common approaches for pattern recognition are template matching, statistical classification, syntactic matching, and computational intelligence classification. A brief description of these approaches is given below.

2.5.1 Template matching

In this approach, a template or prototype is available. The data to be recognised are matched against a prototype while taking into account all allowable positions (translations and/or rotations) or rotation changes. If the resulting distance is smaller than a certain threshold, the pattern is correctly classified. This approach will fail if there are large intra-class variations among the patterns [38].

2.5.2 Statistical methods

In this case, each pattern is represented in terms of "p" features, and it is viewed as a point in a p-dimensional space. The objective, in this approach, is to recognise the patterns based on decision boundaries that are determined by the probability distribution of the patterns belonging to each class. This probability distribution must be either specified or learned [28].

2.5.3 Structural or syntactic methods

In this approach, the patterns are composed of sub-patterns, which are themselves built from simpler sub-patterns. The simplest sub-patterns to be recognized are known as

primitives. The classification, in this approach, is done based on a determined acceptance error in accordance to some predefined rules. The importance of this method lies in the fact that it provides a description of how the given pattern is constructed in terms of the predefined primitives [67].

2.5.4 Computational intelligence

James C. Bezdek presented the term computational intelligence in 1992 [7]. This term describe methods of computation which adapt solutions to new problems, deal only with numerical data, have a pattern recognition component, and do not rely on explicit human knowledge [7], [27]. Computational intelligence emerges because of synergy of genetic, fuzzy, rough, and neural computing. Thanks to this synergy, computational intelligence offers a well-developed algorithmic framework needed to solve problems related to complex system behaviour.

2.6 Data sets to be studied

In this thesis, several data sets were used to evaluate the performance of the different classifiers that were developed. A description of these data sets is given below.

2.6.1 Synthetic data sets

The patterns in the following two data sets were separated into two classes. The patterns for each class were distributed according to a normal density function with unit covariance matrices. The mean vectors (m_1 for class 1 and m_2 for class 2) were:

- For data (a): m_1=(2.0, 2.0) and m_2=(-2.0, -2.0). See Figure 2-2.
- For data (b): m_1=(1.0, 1.0) and m_2=(-1.0, -1.0). See Figure 2-3.

Each class has 50 elements, giving a data set of 100 elements, 60 of which were used for training and 40 for testing.

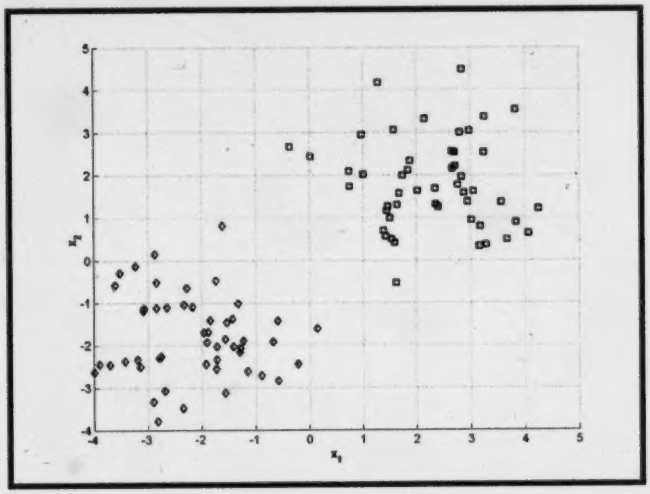


Figure 2-2: Synthetic data (a). Squares and diamonds represent class 1 and class 2 respectively.

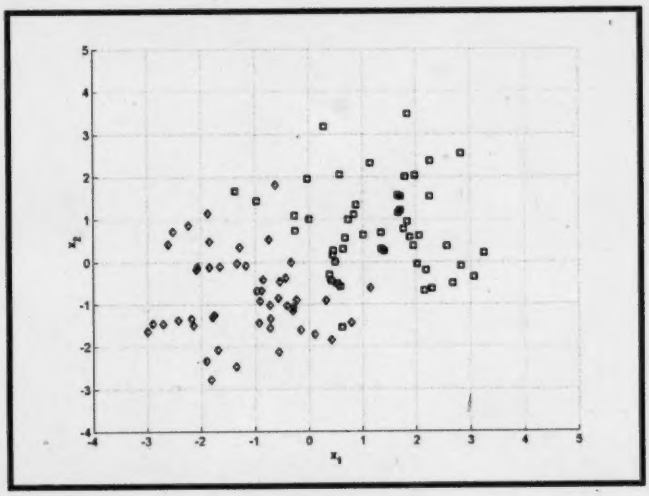


Figure 2-3: Synthetic data (b). Squares and diamonds represent class 1 and class 2 respectively.

The patterns in the following data set were separated into five clusters of 200 elements each. The patterns for each cluster were distributed according to a normal density function with unit covariance matrices. The mean vectors (m_1 for cluster 1, m_2 for cluster 2, m_3 for cluster 3, m_4 for cluster 4, and m_5 for cluster 5) were m_1=(0, 0, 0), m_2=(5, 0, 0), m_3=(0, 5, 0), m_4=(0, 0, 5), and m_5=(5, 5, 5) (see Figure 2-4).

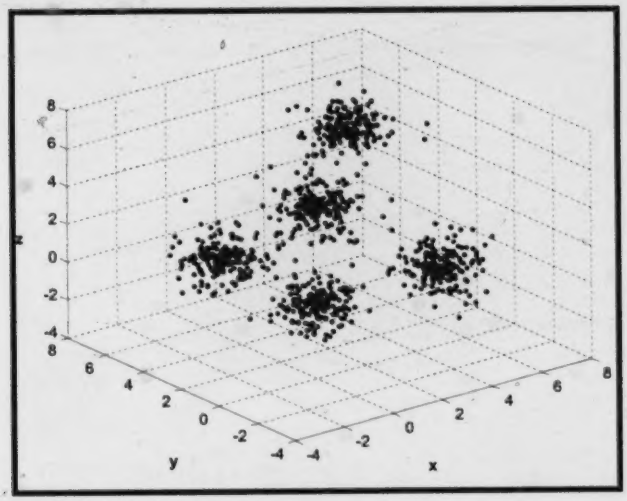


Figure 2-4: Synthetic data (c). Five clusters with unit covariance matrices.

2.6.2 Iris data set

The iris data set first appeared in [2]. This data set consist of patterns representing three varieties of irises: Setosa, Versicolour, and Virginica. Each class has 50 patterns. Each pattern has four attributes indicated in Table 2-3. Figure 2-5 shows a plot of sepal length vs. sepal width. Figure 2-6 shows a plot of petal length vs. petal width. From these figures, it seems that petal length and petal width are the most discriminant features for the iris data set.

The version of the iris data set used here comes from the repository of machine learning data sets at University of California, Irvine[1] and corresponds to the iris data version used by Bezdek et al. [8]. The entire data set includes 150 instances, 75 of which were used for training and 75 for testing.

Number	Attribute	Units
1	Sepal Length	cm
2	Sepal Width	cm
3	Petal Length	cm
4	Petal Width	cm

Table 2-3: Listing of features for the iris data set.

[1] UCI Repository of Machine Learning Databases, C. Blake, E. Keogh, and C. J. Merz. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

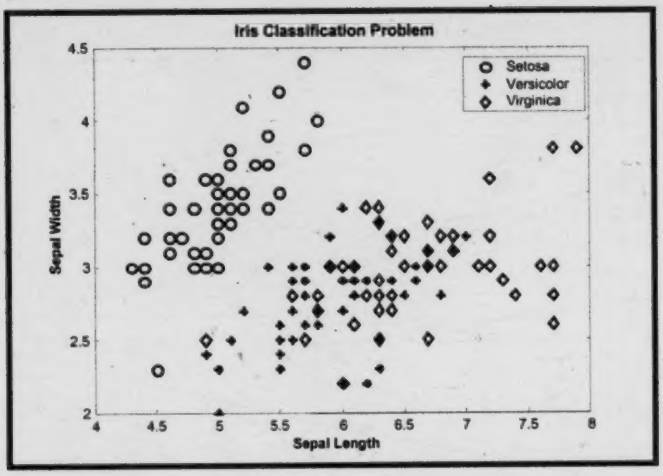


Figure 2-5: A two-dimensional (dimension 1 vs. dimension 2) section of the Anderson iris data.

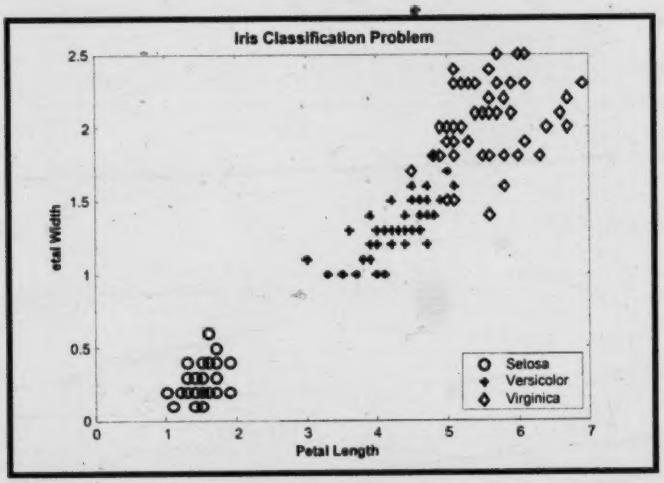


Figure 2-6: A two-dimensional (dimension 3 vs. dimension 4) section of the Anderson iris data.

2.6.3 Wisconsin breast cancer database

Dr. William H. Goldberg created the Wisconsin breast cancer database as part of a study that aimed to diagnose cancer via linear programming [45]. The data set has 11 attributes indicated in Table 2-4. These variables are used to determine whether a tumour is benign or not.

Number	Attribute	Domain
1	Sample code number	Id-number
2	Clump thickness	1-10
3	Uniformity of cell size	1-10
4	Uniformity of cell shape	1-10
5	Marginal adhesion	1-10
6	Single epithelial cell size	1-10
7	Bare nuclei	1-10
8	Bland chromatin	1-10
9	Normal nucleoli	1-10
10	Mitoses	1-10
11	Class	2 for benign and 4 for malign

Table 2-4: Listing of features for the Wisconsin breast cancer database.

The version of the Wisconsin breast cancer database used here comes from the repository of machine learning data sets at University of California, Irvine. The entire data set includes 683 instances, of which 444 were confirmed to correspond to benign tumours and 239 of which were labelled as malign tumours.

2.6.4 Boston housing data set

The Boston housing data set first appeared in [31] and has been studied, among others, in [5], [13], and [70]. It is concerned with house prices in the suburbs of Boston. This data set has 13 continuous attributes and one binary-valued attribute. See Table 2-5 for a description of all the attributes. The classification problem consists of determining whether a house price is greater than or equal to $21,000.

The version of the Boston data set used here comes from the repository of machine learning data sets at University of California, Irvine. The entire data set includes 506 instances, of which 260 correspond to house prices greater than or equal to $21,000 and 246 of which correspond to house prices less than $21,000.

Feature	Information
CRIM	Per capita crime rate by town.
ZN	Proportion of residential land zoned for lots over 25,000 sq. ft.
INDUS	Proportion of non-retail business acres per town.
CHAS	Charles River dummy variable (=1 if tract bounds river; 0 otherwise).
NOX	Nitric oxides concentration (parts per 10 million).
RM	Average number of rooms per dwelling.
AGE	Proportion of owner-occupied units built prior to 1940.
DIS	Weighted distances to five Boston employment centres.
RAD	Index of accessibility to radial highways.
TAX	Full-value property – tax rate per $10,0000.
PTRATIO	Pupil-teacher ratio by town.
B	$1000(Bk- 0.63)^2$ where Bk is the proportion of blacks by town
LSTAT	% Lower status of the population.
MEDV	Median value of owner-occupied homes in $1000's.

Table 2-5: Listing of features for the Boston housing data set.

2.6.5 A Severe storm cell data set

Storm cells, which are the cause of most severe summer weather, may be defined as volumetric patterns that present high radar reflectivity at mid to high altitude levels and exhibit a single updraft [74]. There are four main types of storm cells: hail, rain, tornado, and wind. To characterize them, 22 products (see Table 2-6) [63] (derived from reflectivity values and certain heuristics) might be used. The system used to produce these derived features is the Radar Decision Support System (RDSS) [74], developed for Environment Canada by Infomagnetics Technologies Corporation. A comprehensive description of the algorithms used for the derivation of the 22 previously mentioned and other products may be found in [74].

From a careful analysis of the data, it can be seen that there are a considerable number of identical cells (that is, the 22 features used to describe them have the same values) with two different labels. This happens for about 25% of the cells. To cover those cells with two labels, one may envision adding six more classes to the four original ones (hail, rain, tornado, wind). The new classes are: hail or rain, hail or tornado, hail or wind, rain or tornado, hail or wind, and tornado or wind. The entire data set includes 577 instances, 346 of which were used for training and 231 for testing.

Derived Feature	Values
Height offset or Z value	Real +
Extent (X and Y values)	Integer + and Integer +
Core volume	Integer +
Core height	Integer +
Core size X and Y	Real + and Real +
Core tilt vector X, Y and Z	Real, Real and Real +
Core tilt angle	Real +
Velocity X, Y and Z	Real, Real and Real
Orientation	Real +
Join count	{0, 2}
Split count	{0, 1}
Super cell severity	{0, 1, 2, 3}
Wind gust severity	{0, 1, 2, 3}
Hail occurrence (%)	{0, 50, 90}
Velocity set flag	{0, 1}
Super cell flag	{0, 1}

Table 2-6: Listing of derived features for the severe storm cell data set.

2.7 Summary

This chapter has provided a succinct overview to the problem that is studied in this thesis (i.e. the development of pattern recognition systems, based on the computational intelligence approach, which helps in making decisions automatically and reliably). This chapter has given a brief description of what one of the main tasks of pattern recognition is (i.e. finding structure in a data set to perform classification), as well as a list of examples of pattern recognition applications. The typical pattern recognition system was presented as well as the methods for measuring its performance. Some common approaches to pattern recognition were also presented. Finally, the data sets used in the experiments were described.

The next chapter will present granular computing, a computational intelligence approach that allows for the representation and processing of knowledge.

3 Granular Computing

3.1 Introduction

Granular computing is a computing paradigm that is concerned with the formation and processing of information granules. An information granule is a set of objects that are grouped based on their similarity, proximity or functionality. This chapter begins with a brief introduction to granular computing. A method for generating information granules is then presented. Next, the use of prototype stability analysis to determine the best group of centres to be used in classifiers design is described. Finally, some illustrative numerical examples are presented.

3.2 Information granulation

Granular computing is oriented toward the representation and processing of knowledge related to problems in basic entities known as information granules. An information granule is a set of objects that are grouped based on their similarity, proximity or functionality (see Figure 3-1). These information granules may be crisp (i.e. with sharp boundaries) or fuzzy (i.e. with boundaries that are not well-defined) (See Figure 3-2). The importance of granular computing lies in the fact that a suitable granulation may facilitate the identification of useful patterns in a data set.

Figure 3-1: A granule is a set of objects that are grouped based on their similarity, proximity or functionality.

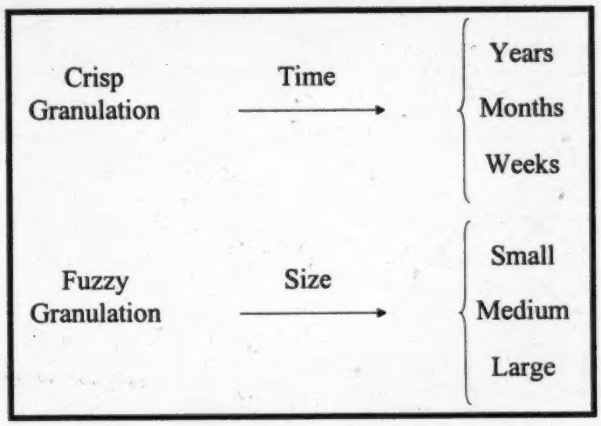


Figure 3-2: Examples of crisp and fuzzy granulation

Information granulation (IG) of an object results in a collection of information granules [79]. For example, granulation of the object "tree" may result in the following granules: roots, trunk, leaves, flowers, and seeds.

IG in which granules are crisp has been used in a broad range of applications including the following: quantization, analog-to-digital-conversion, image segmentation, and many others [80]. Even though crisp information granulation (crisp IG) is very important in these applications, it does not take advantage of the fact that human beings are able to make rational decisions based on partial knowledge and imprecision. For example, in the case of a tree, the boundaries of the roots, trunk, etc. are fuzzy. Therefore, the resulting information granules are fuzzy. Moreover, the granules are associated with fuzzy attributes, e.g., size, colour, and texture in the case of a leaf. Finally, granule attributes have fuzzy values, e.g., in the case of the fuzzy attribute size (of a leaf), the fuzzy values might be small, medium or large. [79] (Figure 3-3).

To deal with imprecise real world situations, Zadeh introduced the concept of fuzzy information granulation (fuzzy IG) [78]-[80]. Fuzzy IG is realized through the use of fuzzy sets, which were introduced by Zadeh in 1973 [77]. Fuzzy sets use symbols (linguistic terms such as *long, fast, very short, around*, etc.) to summarize the domain knowledge in a convenient format [55]. Fuzzy sets are able to express the notion of partial membership of an element to a particular granule [34]. This property is intensively used in clustering algorithms and will be studied in the next section.

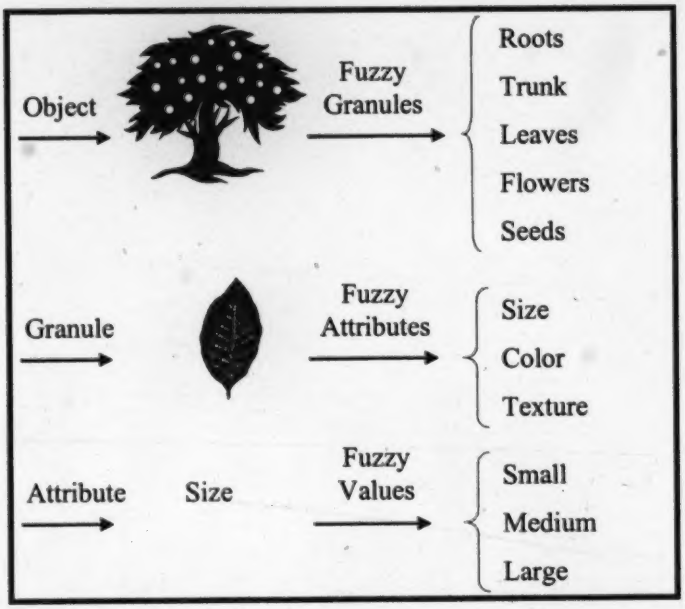


Figure 3-3: Basic structure of fuzzy information granulation: granules, attributes, and values.

3.3 Construction of information granules

Information granulation can be obtained in various ways depending on the type of problem and the type of data available. For example, granules can be obtained manually through expert interviews or automatically by clustering techniques. Expert interviews are useful when the designer wants to obtain information granules that reflect subjective perceptions about concepts associated with the problem that is being solved. In contrast, clustering techniques are used when the information granules must account for information contained in experimental data. This section of the thesis is concerned with the use of clustering techniques (in particular, partitional clustering techniques) for the generation of information granules.

Partitional clustering techniques attempt to find partitions in a sample data set by minimizing the within-cluster dispersion or by maximizing the between-cluster dispersion [38]. So the problem of partitional clustering can be formulated in the following way: given n patterns in an p-dimensional space, determine a partition U of the patterns into c clusters such that the patterns in a cluster are more similar to each other than to patterns of different clusters [38]. To get reliable results, the number of pattern available n should be greater than the

expected number of clusters c. Partitional clustering techniques are also known as objective function methods because for each c, a criterion or objective function measures the "desirability" of clustering candidates [6]. The most widely used objective function model for partitional clustering is based on the weighted-within-groups sum of squared errors. The general goal is to obtain that partition which, for a fixed number of clusters, minimizes the square error.

Two of the most representative partitional clustering techniques are K-means [23] and the fuzzy c-means [6]. The partitional clustering technique to be studied in this thesis is the fuzzy c-means because it can be used to generate fuzzy information granules that are required for the development of a classifier presented in Chapter 6. The fuzzy c-means (FCM) overcomes the problem of assigning a particular class to those patterns that are doubtful, using the fuzzy set concept of intermediate membership degree [50]. Therefore, the jth pattern belongs to the ith cluster to a degree specified by a membership grade "u_{ij}". For this reason, the partition matrix **U**, which is often called the fuzzy partition matrix, should satisfy these conditions [10]:

$$0 \le u_{ij} \le 1, \quad \text{for all} \quad i = 1,2,...,c \quad j = 1,2,...,n \tag{3-1}$$

$$\sum_{i=1}^{c} u_{ij} = 1, \quad \text{for all} \quad j = 1,2,...,n \tag{3-2}$$

$$0 < \sum_{j=1}^{n} u_{ij} < n, \quad \text{for all} \quad i = 1,2,...,c \tag{3-3}$$

and the objective function assumes the following form [10]:

$$Q = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m} d_{ij}^{2} \tag{3-4}$$

where

$$d_{ij} = \left\| \mathbf{x}_j - \mathbf{v}_i \right\|_l = \left[\sum_{s=1}^{p} | x_{js} - v_{is} |^l \right]^{(1/l)}, \quad l > 1$$

(3-5)

with $\mathbf{v}_i = (v_{i1}, v_{i2}, ..., v_{ip})$ being a vector that corresponds to a p-dimensional (unknown) cluster centre (weight, or prototype); and d_{ij} is normally the Euclidean norm, but in general it is the Minkowski distance or l-norm distance that is often used with the Hamming or Manhattan (l=1), Euclidean (l=2) and Tschebyschev (l= ∞) distances [56].

The iterative minimization of the objective function (for m>1) involves the calculation of the partition matrix and the prototypes in the following way [49]:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(d_{ij} / d_{kj} \right)^{2/(m-1)}}, \quad 1 \le i \le c; 1 \le j \le n$$

(3-6)

$$\mathbf{v}_i = \frac{\sum_{j=1}^{n} u_{ij}^m \mathbf{x}_j}{\sum_{j=1}^{n} u_{ij}^m}, \quad 1 \le i \le c$$

(3-7)

The FCM method has two essential design components:

1. The distance function d_{ij} that will affect the shape of the clusters that are produced. As shown in [10], in the $\Re^2$ space, for the Manhattan distance, the clusters will have a diamond shape, for the Euclidean distance, the shape will be circular, and for the Tschebyschev distance, the clusters will be squares. The distance function has also a significant impact in the calculations of the prototypes because differentiation of d_{ij} in (3-5) leads to the different forms of prototypes **v**. When d_{ij} is defined by the Euclidean distance, the resulting prototypes can be interpreted as the weighted (fuzzy) means of the clusters (3-7). When d_{ij} is defined by a non-differentiable Minkowski metric, the calculation of prototypes becomes more complicated. In this case, a method such as linear programming or Newton's method may be required to estimate **v** numerically. In the particular

case of the Manhattan distance, the calculation of v can be interpreted as the construction of the weighted (fuzzy) medians of the clusters.

2. The fuzzification parameter (m) that affects the form of the membership functions being produced (3-6). For increasing values of "m", there is a deep rippling effect where the membership functions tend to exhibit more local minima. Additionally, for $m \to \infty$ we have that $u_{ij} \to 1/c$. For lower values of "m", the membership functions look like the characteristic functions of sets; so fewer elements with intermediate membership values are obtained [56]. In the limit, when $m \to 1$, the prototypes given by (3-7) resemble the means of the clusters. It is important to mention that the best choice for "m" is probably in the interval [1.5, 2.5] [49] whose midpoint "m=2" has often been the preferred choice for many users of FCM. This is mainly because it constitutes a reasonable compromise between membership functions with excessive oscillations in the membership values and set-like membership functions [56].

The FCM Algorithm [6] is carried out as a sequence of the following steps:

1. Fix the number of clusters c such that c is greater than 2 and less than the number of data samples. Choose the form of the distance function to be used (Manhattan, Euclidean or Tschebyschev). Fix the fuzzification factor such that $m \in (1, \infty)$. Initialize the fuzzy partition matrix.
2. Calculate the c fuzzy prototypes $\mathbf{v}_i$, $1 \le i \le c$.
3. Update the fuzzy partition matrix using the just calculated prototypes.
4. Compare the actual partition matrix with the previous one. If they are close enough (i.e., $\left\| \mathbf{U}^{actual} - \mathbf{U}^{old} \right\| = \max\left(\left| u_{ij}^{actual} - u_{ij}^{old} \right| \right) < \varepsilon$, where ε is a threshold typically in the interval [0.00001, 0.01]) then stop.
5. Go to step 2.

3.4 Prototypes stability analysis to determine an appropriate number of information granules to be used in classifiers

One of the most important considerations before applying a partitional clustering technique is the selection of the number of clusters (c). An error in the choice of the number of

clusters may prevent for a correct detection of clustering structure [24]. This can be illustrated by considering the data set depicted in Figure 3-4. If clustering is applied to ten, randomly selected, portions of the data set, the following results may be obtained: for c=5, the resulting prototypes for each cluster are very close to each other (see Figure 3-5); while for c=4, some of the prototypes are split into two visually different clusters (see Figure 3-6). From Figures 3-5 and 3-6, we conclude that if the number of clusters is appropriate, any prominent data structure ought to survive even if clustering is applied only to a random portion of the data. This idea will be used in the development of a method to determine the appropriate number of clusters for clustering a data set. This method is explained below.

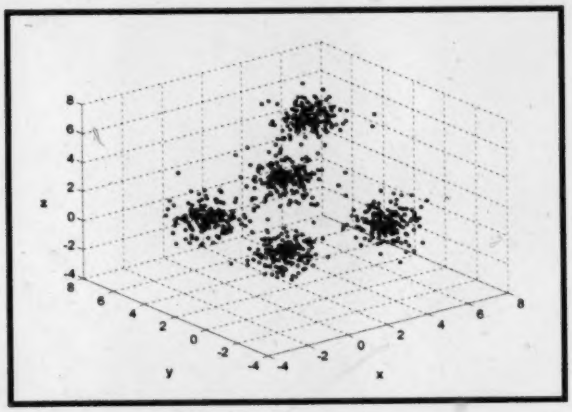


Figure 3-4: Synthetic data (c).

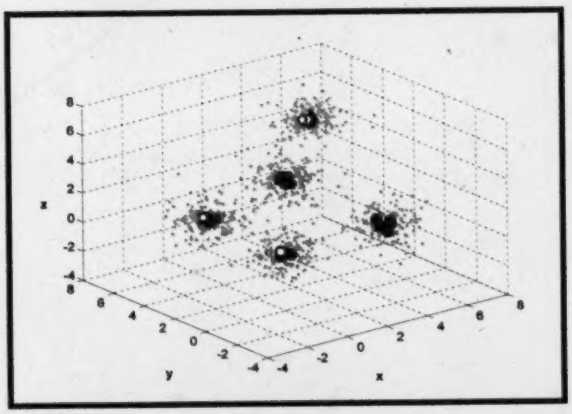


Figure 3-5: Prototypes generated by the FCM with c=5.

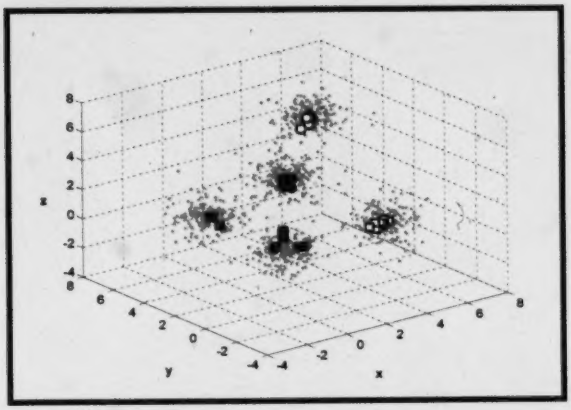


Figure 3-6: Prototypes generated by the FCM with c=4.

Let us start with the results of clustering collected in the matrix of prototypes $\mathbf{v} \in \Re^{c \times p}$ (with c being the number of clusters and p being the number of dimensions). Then, the data set is analyzed by clustering n_p random portions of the data and n_p matrices $\mathbf{v}^1, \mathbf{v}^2, ..., \mathbf{v}^k, ..., \mathbf{v}^{n_p}$ are obtained.

The first step, which has to be performed to do the prototypes stability (or variability) analysis, is concerned with identification of the corresponding prototypes in each matrix $\mathbf{v}^k$. In this thesis, a suboptimal[2] way to determine the relationship between prototypes is applied. It consists of the following steps:

a. Initialize the matrix index (k) and the cluster index (i) to one, i.e., k=1, i=1.

b. Compare the ith prototype of the matrix $\mathbf{v}^k$ with the prototypes of the matrix $\mathbf{v}^{k+1}$ and find an index j_o, j_o=1,2,...,c, for which the distance between the ith prototype of the matrix $\mathbf{v}^k$ and the j_oth prototype of the matrix $\mathbf{v}^{k+1}$ attains a minimum.

c. Perform step "b" for all i, i=1,2,...,c to determine a correspondence between the prototypes of the matrix $\mathbf{v}^k$ and $\mathbf{v}^{k+1}$.

d. Reset i (i.e., i=1) and increase k (i.e., k=k+1) if $k<n_p$ go to step "b" otherwise calculate the standard deviation $\boldsymbol{\sigma} \in \Re^{c \times p}$ of the corresponding prototypes by

---

[2] It is suboptimal because the relationship between prototypes is determined based on a specified distance measure (in this thesis we used Euclidean distance). Different distance measures may lead to different results. Finally, to be optimal, the real assignation of prototypes to clusters must be known.

$$\sigma_{ij} = \sqrt{\frac{1}{n_p - 1}\sum_{k=1}^{n_p}\left(v_{ij}^{k} - \bar{v}_{ij}\right)^2} \quad , \quad i = 1,2,...,c \ , \ j = 1,2,...,p$$

(3-8)

where c is the number of clusters, p is the number of dimensions of the data, n_p is the number of portions in which the data set was divided, v_{ij}^{k} is the value of the jth dimension of the prototype corresponding to the ith cluster for the kth portion of the data, and $\bar{v}_{ij}$ is the average of the jth dimension for the prototypes of the ith cluster, that is:

$$\bar{v}_{ij} = \frac{1}{n_p}\sum_{k=1}^{n_p} v_{ij}^{k} \ , \quad i = 1,2,...,c \ , \ j = 1,2,...,p$$

(3-9)

The above-mentioned method allows us to arrange the results of clustering in the following tabular form

	Data Portions			
Clusters	1	2	...	n_p
1	$\mathbf{v}_1^1$	$\mathbf{v}_1^2$	...	$\mathbf{v}_1^{n_p}$
2	$\mathbf{v}_2^1$	$\mathbf{v}_2^2$	...	$\mathbf{v}_2^{n_p}$
...	...	...	...	...
c	$\mathbf{v}_c^1$	$\mathbf{v}_c^2$	...	$\mathbf{v}_c^{n_p}$

Table 3-1: Results obtained by clustering the n_p portions of the data.

Every column in Table 3-1 corresponds to the representation of the ith cluster obtained by applying a clustering algorithm on the kth portion of the data. With these results, the average scattering (or stability index) per dimension for c clusters can be calculated by

$$\tau_{cj} = \frac{\frac{1}{c}\sum_{i=1}^{c}\sigma_{ij}}{r_j} \ , \ 1 \le j \le p \ , \ 2 \le c \le c_{max}$$

(3-10)

where p is the number of dimensions, c_{max} is the maximum number of clusters that are tested, σ_{ij} is the standard deviation of the jth dimension for the prototypes of the ith cluster, and r_j is the range per dimension of the training data ($\mathbf{X}$) given by

$$\mathbf{r} = \boldsymbol{max}(\mathbf{X}) - \boldsymbol{min}(\mathbf{X}) \ , \ \mathbf{r} \in \Re^p$$

(3-11)

The average scattering indicates the average of the variation within the clusters for a number of clusters set to c. A small value for this term indicates a partition where the prototypes in each cluster are close to each other. As the scattering within the clusters increase, the prototypes are getting farther away. Bearing this in mind, a number of clusters which minimizes the scattering index can be considered as an optimal value to be used for clustering the data. Therefore, the appropriate number of clusters per dimension (nc_j) is

$$nc_j = \boldsymbol{arg}\left[\boldsymbol{min}_{c=2,3,\ldots c_{max}} \tau_{cj}\right] \ , \ 1 \le j \le p$$

(3-12)

Now, if all the dimensions are going to be calculated at once, the number of clusters can be determined as a function of the number of clusters per dimensions, that is

$$nc = \boldsymbol{f}\left(nc_j\right)$$

(3-13)

In this thesis, nc is obtained by rounding the mean of the number of clusters per dimension to the nearest integer greater than or equal to this mean.

The overall algorithm for applying prototypes stability analysis (PSA) to determine the appropriate number of clusters is given by the following steps [65]:

Given	The data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \Re^p$, a set of n feature vectors in a p-dimensional space.
Defined	c_{max}, the maximum number of clusters. n_p, the number of portions in which the data set will be randomly divided. Any parameter needed for the clustering algorithm at hand.
Initialization	Randomly divide the data set in n_p non-overlapping subsets.
Processing	For each number of clusters $c \in \{2,3,\ldots,c_{max}\}$ a. Apply clustering to each one of the n_p subsets and sort and record their prototypes. b. Calculate the standard deviation per dimension of the n_p prototypes in each one of the clusters. c. Calculate the stability coefficient by using: $\tau_{cj} = \frac{\frac{1}{c}\sum_{i=1}^{c}\sigma_{ij}}{r_j}$, $1 \le j \le p$, $2 \le c \le c_{max}$ with r_j being the range per dimension. Determine the appropriate number of clusters (per dimension) by: $nc_j = \boldsymbol{arg}\left[\boldsymbol{min}_{c=2,3,\ldots,c_{max}} \tau_{cj}\right]$, $1 \le j \le p$ Determine the number of clusters to be used in modeling by calculating the mean of the number of clusters per dimension $nc_{average} = \frac{1}{p}\sum_{j=1}^{p} nc_j$ and rounding $nc_{average}$ to the nearest integer greater than or equal to $nc_{average}$.
Result	Number of clusters, prototypes or information granules to be used in the design of classifiers.

Table 3-2: Algorithm for applying PSA.

Summarizing PSA, the original data set is randomly re-sampled to produce n_p subsets. Clustering is applied to each subset. The resulting prototypes are sorted and the stability index (3-10) is determined. The number of clusters that minimizes the stability index is chosen as the number of clusters in the data. PSA will favour those partitions that consist of clusters that are stable against re-sampling. Moreover, if there is overlap between clusters (as depicted in Figure 3-7), PSA may prefer a number of clusters greater than the true number of clusters to produce prototypes that are robust against re-sampling (for instance, in the case of Figure 3-7, PSA may prefer 3 as the number of clusters instead of 2). Finally, if the data set is so sparse

that re-sampling can remove the underlying structure, PSA should not be used to determine the appropriate number of clusters.

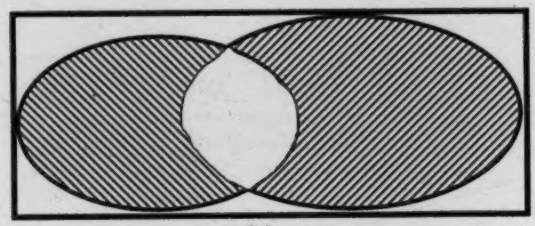


Figure 3-7: Two overlapping clusters.

3.5 Illustrative examples

These studies rely on three synthetic data sets and a widely available data set, the iris data set.

3.5.1 Synthetic data sets

For the first group of experiments, two classes of vectors were generated. The vectors were distributed according to a normal density function with unit covariance matrices. The mean vectors (m_1 for class 1and m_2 for class 2) were:

- For data (a): m_1=(2.0, 2.0) and m_2=(-2.0, -2.0). See Figure 3-8.
- For data (b): m_1=(1.0, 1.0) and m_2=(-1.0, -1.0). See Figure 3-9.

Each class had 50 elements, giving a data set of 100 elements, 60 of which were used for clustering. To obtain reliable results, a rotation method was employed by randomly re-sampling the data to be clustered and repeating the experiments ten times. The results are summarized in Table 3-3 for data (a) and in Table 3-4 for data (b).

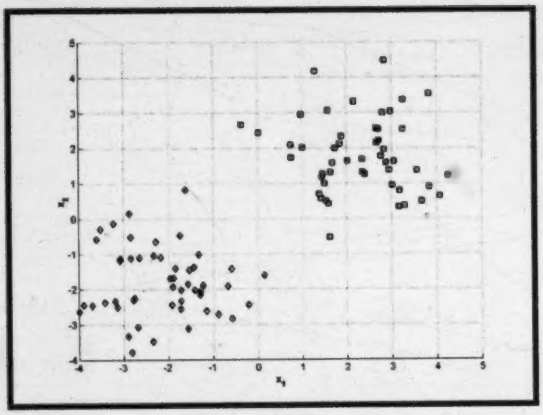


Figure 3-8: Synthetic data (a). Squares and diamonds represent class 1 and class 2 respectively.

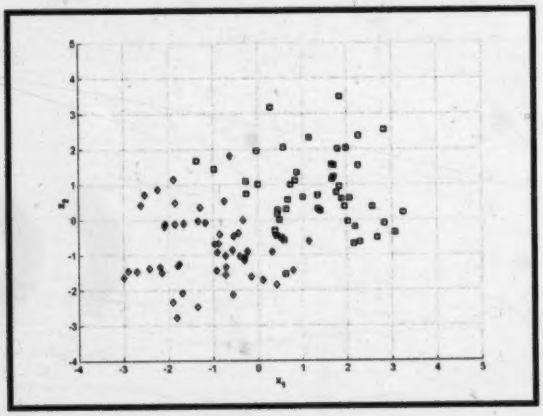


Figure 3-9: Synthetic data (b). Squares and diamonds represent class 1 and class 2 respectively.

	Data Group									
	1	2	3	4	5	6	7	8	9	10
nc_j (dim1/dim2)	2/9	2/8	2/2	9/2	2/2	2/2	2/2	2/2	9/2	2/2
nc	6	5	2	6	2	2	2	2	6	2

Table 3-3: Number of clusters (nc) to be used in clustering data (a).

	Data Group									
	1	2	3	4	5	6	7	8	9	10
nc_j (dim1/dim2)	2/2	2/4	2/8	2/9	9/2	2/2	2/2	2/4	2/8	2/2
nc	2	3	5	6	6	2	2	3	5	2

Table 3-4: Number of clusters (*nc*) to be used in clustering data (b).

For the next group of experiments a data set - data (c) - with five clearly defined clusters was used (see Figure 3-10). The patterns for each cluster were distributed according to a normal density function with unit covariance matrices. The mean vectors (m_1 for cluster 1, m_2 for cluster 2, m_3 for cluster 3, m_4 for cluster 4, and m_5 for cluster 5) were m_1=(0, 0, 0), m_2=(5, 0, 0), m_3=(0, 5, 0), m_4=(0, 0, 5), and m_5=(5, 5, 5) (see Figure 2-4).

Each cluster had 200 elements, giving a data set of 1000 elements, 500 of which were used for clustering. To obtain reliable results, a rotation method was employed by randomly re-sampling the data to be clustered and repeating the experiments ten times. The results are summarized in Table 3-5.

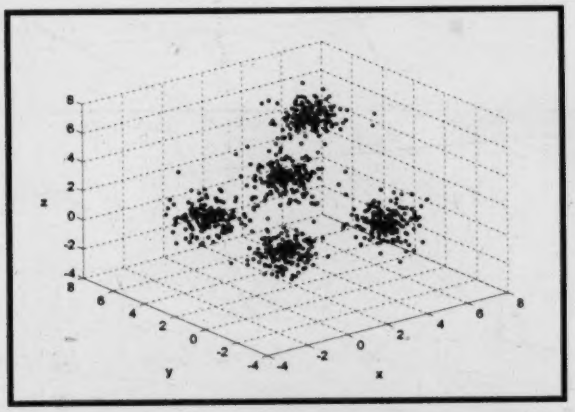


Figure 3-10: Synthetic data (c).

	Data Group									
	1	2	3	4	5	6	7	8	9	10
nc_j (dim1/dim2/dim3)	5/5/5	5/5/5	5/2/7	4/5/4	5/5/5	5/5/5	5/5/5	6/5/5	6/5/6	5/5/5
nc	5	5	5	5	5	5	5	6	6	5

Table 3-5: Number of clusters (*nc*) to be used in clustering data (c).

3.5.2 Iris data set

This data set consist of patterns representing three varieties of irises: Setosa, Versicolour, and Virginica. Each class has 50 patterns, giving a data set of 150 elements, 75 of which were used for clustering. To obtain reliable results, a rotation method was employed by randomly re-sampling the data to be clustered and repeating the experiments ten times. The results are summarized in Table 3-6.

	Data Group									
	1	2	3	4	5	6	7	8	9	10
nc_j (dim1/dim2/ dim3/dim4)	3/6/ 3/6	2/2/ 2/2	2/4/ 2/2	2/3/ 5/4	3/3/ 8/8	2/2/ 2/2	5/3/ 10/10	7/2/ 2/3	2/2/ 3/2	2/2/ 2/4
nc	5	2	3	4	6	2	7	4	3	3

Table 3-6: Number of clusters (nc) to be used in clustering the iris data.

To verify the applicability of PSA in determining a stable structure in the data, the previous results will be used in the design of classifiers later in this thesis.

3.6 Summary

This chapter has provided an overview to granular computing and clustering. This chapter has also described how to use fuzzy clustering to generate information granules. Finally, how to determine an appropriate number of information granules to be used in modeling using prototypes stability analysis was presented.

The next chapter will present one of the most recent computational intelligence developments in pattern recognition, the support vector machine (SVM) approach.

4 Support vector machines (SVM's) in pattern recognition

4.1 Introduction

The support vector machine (SVM) approach [20], [71] is a binary classification technique that has been used with a high degree of success in several applications[3] ranging from bioinformatics to face identification and text categorisation. In this approach, training involves the maximization of the margin, which is the minimum distance from any training point to a separating hyperplane [12]. The location and slope of this hyperplane are defined by a group of training points called support vectors. This chapter presents a brief introduction to the SVM. For further information, the reader is referred to [14], [22], [71], [72]. This chapter begins with three cases that appear when studying the SVM approach. Some illustrative examples are then presented, followed by a description of an algorithm to select the SVM parameters. Next, the extension of the SVM approach to the multi-class classifier is presented. Finally, some implementation aspects are studied.

4.2 Theory

To introduce the subject, one can begin by considering the illustrative binary classification problem shown in Figure 4-1. It can be seen that there are a number of hyperplanes that one can draw to correctly divide the training data into two classes. The hyperplane in Figure 4-1 (a) correctly classifies all the training points, but in the case of a testing point like the bold pattern shown in Figure 4-1 (b), the decision system would misclassify it. Now, in the case of a hyperplane that maximizes the distance from any training pattern to it, as depicted in Figure 4-1 (c), the decision system would correctly classify the bold testing pattern as shown in Figure 4-1 (d). This hyperplane, which correctly classifies all the training patterns and maximizes the margin (i.e. the minimum distance from any training pattern to the hyperplane), is called the optimal separating hyperplane (OSH). The optimal separating hyperplane is the key concept of the maximal margin classifier, which is the base architecture of the SVM approach.

---

[3] See http://www.clopinet.com/isabelle/Projects/SVM/applist.html

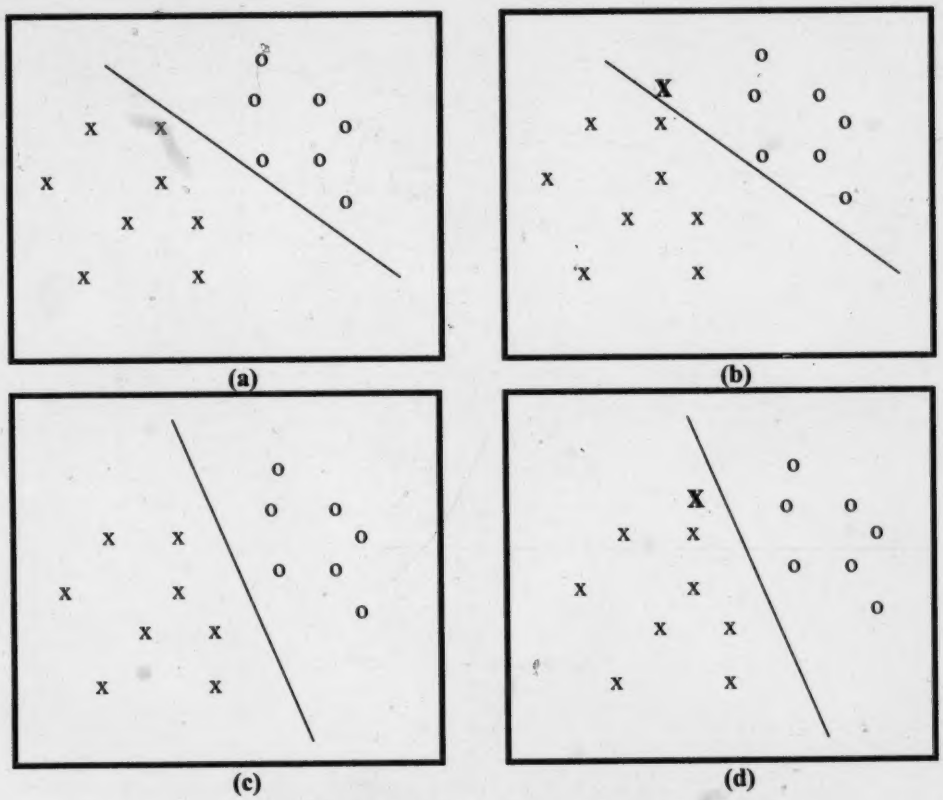


Figure 4-1: Illustrative classification problem.

4.2.1 The maximal margin classifier

The maximal margin classifier is the simplest model of the Support Vector Machine. It works only for data sets that are linearly separable as is the one depicted in Figure 4-2. From a linearly separable data set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \Re^p$, $y_i \in \{-1, +1\}$ (see Figure 4-2) one can intuitively derive several separating hyperplanes that satisfy

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n \tag{4-1}$$

where $\mathbf{w} \in \Re^p$ is a p-dimensional vector that corresponds to the slope of the hyperplane, b is a bias or threshold, and $\mathbf{w} \cdot \mathbf{x}_i$ is a dot product of vectors $\mathbf{w}$ and $\mathbf{x}_i$. The dot product is given by

$$\mathbf{w} \cdot \mathbf{x}_i \equiv \sum_{j=1}^{p} w_j x_{ij}$$

(4-2)

with p being the number of dimensions of the input data.

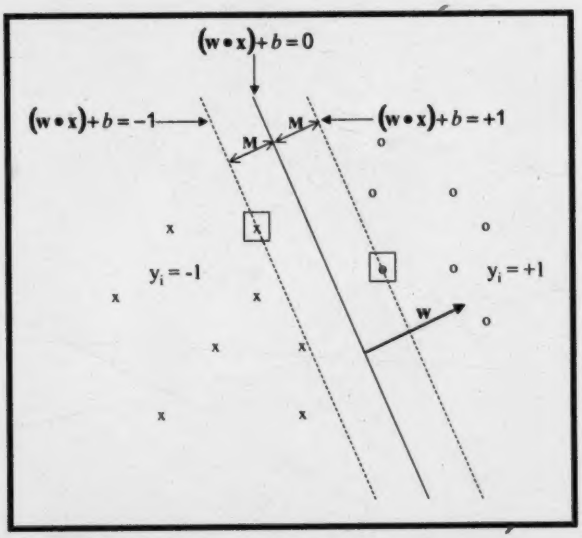


Figure 4-2: An optimal margin classifier. The dashed lines identify the margin $M=1/\|\mathbf{w}\|$. The patterns inside the squares are the support vectors.

An optimal separating hyperplane is one that satisfies the condition (4-1) and minimizes the function

$$\tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

(4-3)

with respect to the vector w and the scalar b. $\|\mathbf{w}\|$ represents the length of the vector w. This length is given by

$$\|\mathbf{w}\| = \sqrt{\mathbf{w} \cdot \mathbf{w}}$$

(4-4)

It is important to mention that the minimization of (4-3) corresponds to the maximization of the margin

$$M=1/\|\mathbf{w}\|$$

(4-5)

The solution to the problem stated in (4-3) is obtained by finding the saddle points of the Lagrange function

$$L(\mathbf{w},b,\boldsymbol{\alpha})=\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{n}\alpha_i\{[(\mathbf{x}_i\cdot\mathbf{w})+b]y_i-1\}$$

(4-6)

where α_i is the *i*th Lagrange multiplier. The function (4-6) has to be minimized with respect to **w** and *b* and maximized with respect to $\alpha_i \geq 0$.

The dual form of (4-6) is given by the maximization of

$$W(\boldsymbol{\alpha})=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j)$$

(4-7)

subject to

$$\alpha_i \geq 0$$

(4-8)

and

$$\sum_{i=1}^{n} y_i\alpha_i=0$$

(4-9)

The reader is referred to [72], [14] for a detailed derivation.

The classification function is determined by:

$$f(x)=\mathbf{sign}\left(\sum_{i/\mathbf{x}_i\in SV} y_i\alpha_i^{o}(\mathbf{x}_i\cdot\mathbf{x})+b_o\right)$$

(4-10)

where $\mathbf{x}$ is a p-dimensional input, $\mathbf{x}_i$ is the ith input that belongs to the support vector (SV) set, y_i is the class label for the ith input, α_i^o is the Lagrange multiplier associated with the ith input, and b_o is computed as:

$$b_o = -\frac{1}{2}\left[\min_{y_i=1}(\mathbf{w}_o \cdot \mathbf{x}_i) + \max_{y_i=-1}(\mathbf{w}_o \cdot \mathbf{x}_i)\right] \qquad (4\text{-}11)$$

where $\mathbf{x}_i$ is the ith SV, and $\mathbf{w}_o$ is the p-dimensional weighting vector that corresponds to the multidimensional "slope" of the optimal hyper plane. The vector $\mathbf{w}_o$ is computed as:

$$\mathbf{w}_o = \sum_{i=1}^{n} y_i \alpha_i^o \mathbf{x}_i \qquad (4\text{-}12)$$

4.2.2 The linear soft-margin classifier

In the case of data sets that are not linearly separable, as depicted in Figure 4-3, the linear SVM may be generalized by introducing n non-negative variables $\xi = (\xi_1, \ldots, \xi_n)$, one for each training pattern. These variables reflect the amount of misclassification that is equal or greater than zero [61]. In this situation, the constraints for α_i^o take the following form:

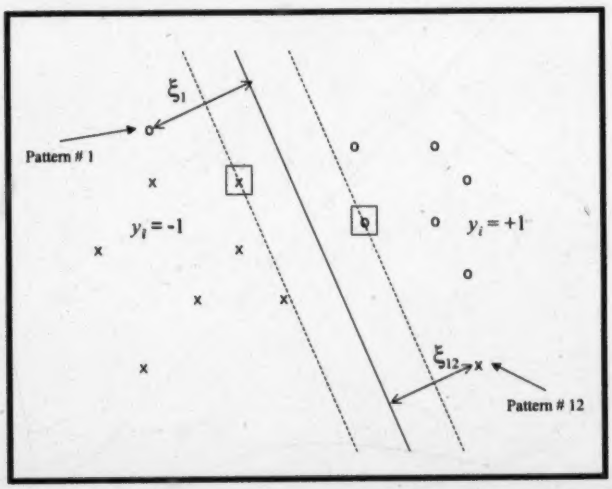


Figure 4-3: A soft margin classifier. The dashed lines identify the margin. The patterns inside the square are the support vectors. ξ_1 and ξ_{12} are the positive variables associated with patterns 1 and 12 respectively. All the other ξ_i are equal to zero.

$$\sum_{i=1}^{n} \alpha_i^0 y_i = 0,\ 0 \le \alpha_i^o \le C,\ i = 1,\ldots,n$$

(4-13)

where C is the error penalty term and n is the number of data patterns.

The classification function is determined by (4-10).

4.2.3 The non-linear classifier

This case, depicted in Figure 4-4, was discussed for the first time in [20]. In this situation, the solution is to map, using the function Φ, the input data to a higher dimensional space where the data become linearly separable. Now, the classification function becomes

$$f(\mathbf{x}) = sign\left(\sum_{i/\mathbf{x}_i \in SV} y_i \alpha_i^0 (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) + b_o \right)$$

(4-14)

with the constraints:

$$\sum_{i=1}^{n} \alpha_i^o y_i = 0,\ 0 \le \alpha_i^o \le C,\ i = 1,\ldots,n$$

(4-15)

To avoid the explicit calculation of the dot product $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})$, a function $K(\mathbf{x}_i, \mathbf{x})$, called the kernel (see Table 4-1), may be used. So the decision function becomes:

$$f(\mathbf{x}) = sign\left(\sum_{i/\mathbf{x}_i \in SV} y_i \alpha_i^o K(\mathbf{x}_i, \mathbf{x}) + b_o \right)$$

(4-16)

where $\mathbf{x}_i$ is the ith *SV* and $\mathbf{x}$ is an input vector.

Kernel	Expression
Linear	$K(\mathbf{x}_i, \mathbf{x}) = (\mathbf{x} \cdot \mathbf{x}_i)$
Polynomial of degree d	$K(\mathbf{x}_i, \mathbf{x}) = (1 + \mathbf{x} \cdot \mathbf{x}_i)^d$
Radial basis function with spread γ	$K(\mathbf{x}_i, \mathbf{x}) = exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\right)$

Table 4-1: Some commonly used kernels

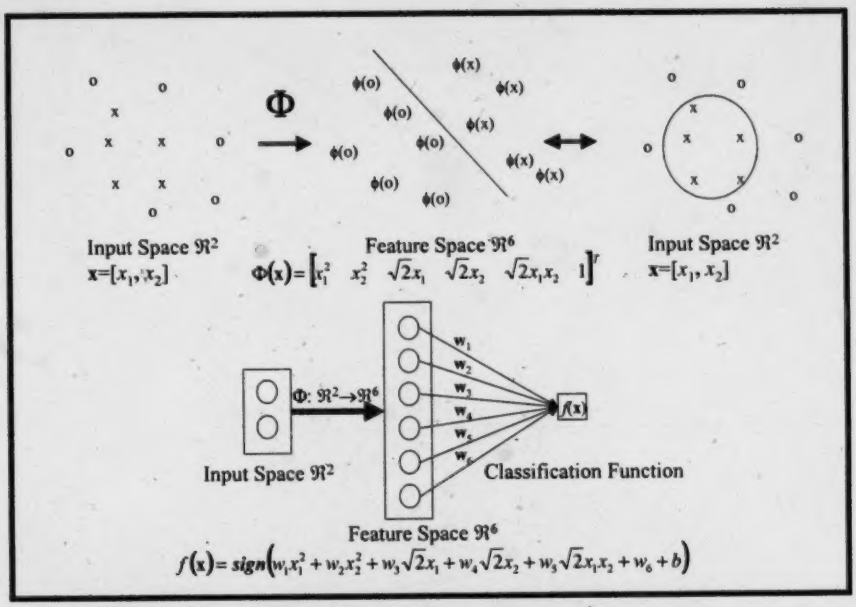


Figure 4-4: A non-linear classifier. The mapping from the input space (top left) to the feature space (top middle) is performed using the function Φ. The separating hyperplane in the feature space (top middle), as determined by the SVM (bottom), corresponds to a non-linear decision surface in the input space (top right).

With an appropriate choice of kernel, the original data may become linearly separable in the feature space even though they are not linearly separable in the input space [15]. The selection of the kernel is still a research issue, although certain approaches to include previous knowledge of the problem have been attempted [69], [4].

Summarizing the SVM approach, the input vector $\mathbf{x}$ and the support vectors $\mathbf{x}_i$ are mapped into a feature space where the necessary dot products are computed by using the kernel function $\boldsymbol{K}$. The results are linearly combined by weights $\lambda_i = y_i\alpha_i$, where y_i is the label of the ith training sample and α_i is its Lagrange multiplier. The kernel function is chosen a priori, and it determines the type of classifier (polynomial or radial basis function). All other parameters (the number of support vectors and weights, and the threshold b) are found during training by solving a quadratic programming problem [68].

4.3 Illustrative examples

For these experiments, two classes of vectors were generated. The vectors were distributed according to a normal density function with unit covariance matrices. The mean vectors were m_1=(2.0, 2.0) and m_2=(-2.0, -2.0). See Figure 4-5.

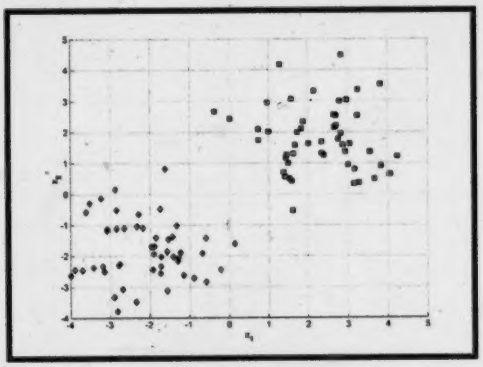


Figure 4-5: Synthetic data. Squares and diamonds represent class 1 and class 2 respectively.

Examples of decision boundaries generated by the SVM approach using the three kernel functions depicted in Table 4-1 are shown in Figures 4-6 through 4-8. From these figures, it is clear that the selection of the kernel affects the shape of the generated decision boundary. This is particularly useful because by just changing the kernel, the SVM technique can be applied to different data distributions.

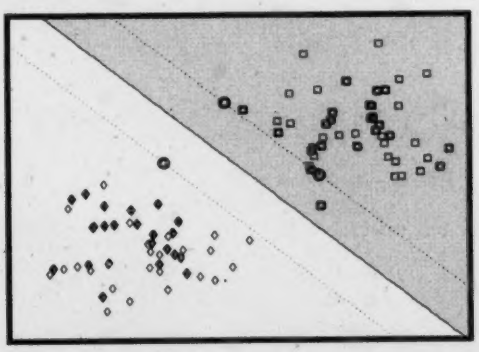


Figure 4-6: The discrimination boundary produced by the SVM with the linear kernel for *C*=100. The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern.

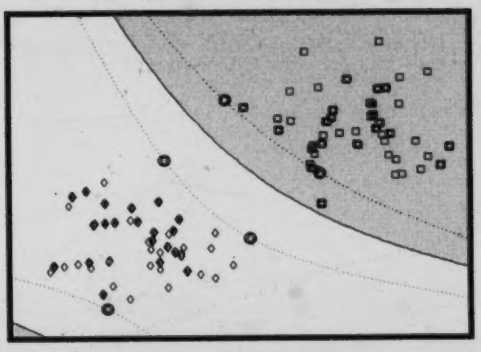


Figure 4-7: The discrimination boundaries produced by the SVM with the polynomial kernel (d=2) for C=100. The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern.

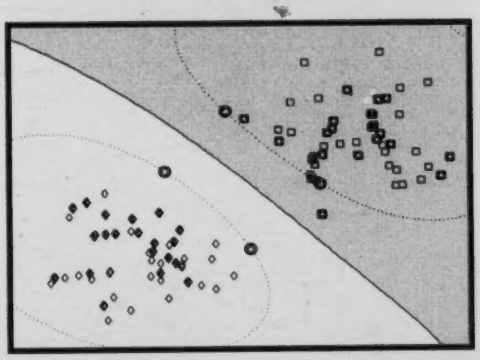


Figure 4-8: The discrimination boundary produced by the SVM with the radial basis function kernel (γ=0.04) for C=100. The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern.

4.4 Model selection

To study the effect of the kernel parameters in the decision boundaries that are generated by the SVM, a group of experiments (with the same data set used in the previous section) was performed. For the linear kernel, the decision boundaries for several values of C are shown in Figure 4-9 (C=0.1 in Figure 4-9 (a), C=1 in Figure 4-9 (b), C=10 in Figure 4-9 (c), and C=1000 in Figure 4-9 (d)). From this figure, it can be seen that the decision boundaries with smaller C (see Figure 4-9 (a) and Figure 4-9 (b)) have a larger margin and a larger number of support vectors, while the solutions with larger C (see Figure 4-9 (c) and

Figure 4-9 (d)) have a smaller number of support vectors. Moreover, there is no significant difference between the decisions boundaries generated by the SVM with the linear kernel for C=10 and C=1000 (see Figure 4-9 (c) and Figure 4-9 (d)).

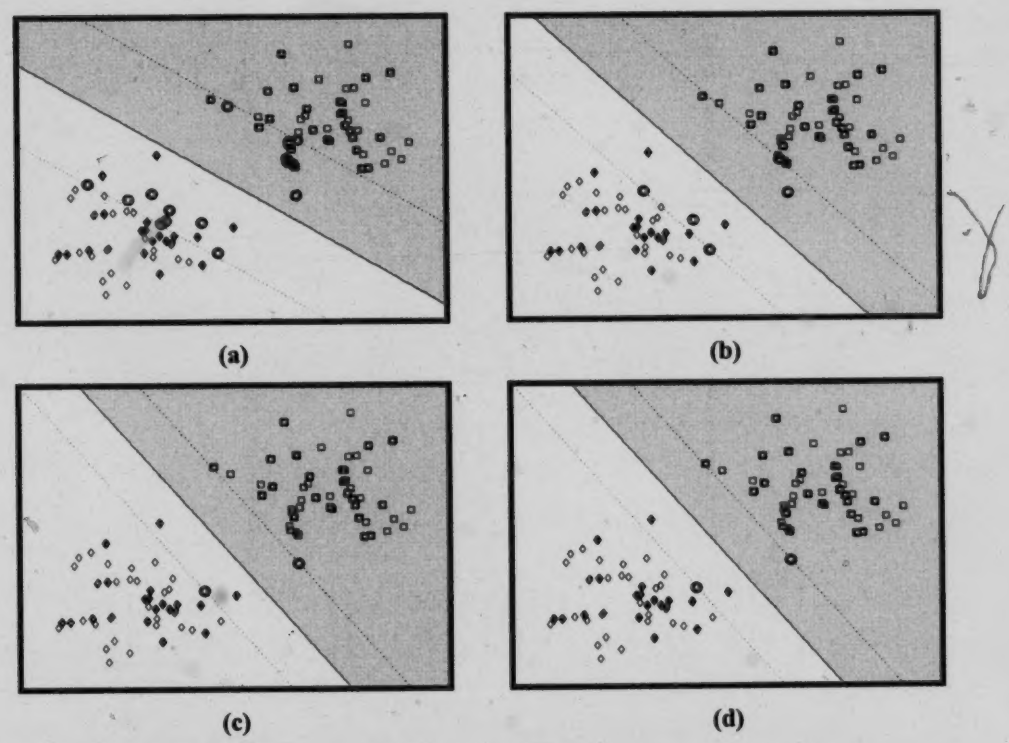


Figure 4-9: Decision boundaries generated by the SVM with the linear kernel for C=0.1 (a), C=1 (b), C=10 (c), and C=1000 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern.

For the polynomial kernel, the decision boundaries for C=100 and several values of d are shown in Figure 4-10 (d=2 in Figure 4-10 (a), d=3 in Figure 4-10 (b), d=5 in Figure 4-10 (c), and d=10 in Figure 4-10 (d)). From this figure, it can be seen that the decision boundaries change when the value of d is changed (see Figure 4-10 (a) and Figure 4-10 (d)). On the other hand, the number of support vectors was the same for the four values of d that were tested.

For the polynomial kernel, the decision boundaries for d=2 and several values of C are shown in Figure 4-11 (C=0.1 in Figure 4-11 (a), C=1 in Figure 4-11 (b), C=10 in Figure 4-11 (c), and C=1000 in Figure 4-11 (d)). From this figure, it can be seen that the decision boundaries with smaller C (see Figure 4-11 (a)) have a larger margin and a larger number of support vectors, while the solutions with larger C (see Figure 4-11 (d)) have a smaller number of support vectors. Moreover, there is not significant difference between the decisions boundaries generated for C=10 and C=1000 (see Figure 4-11 (c) and Figure 4-11 (d)).

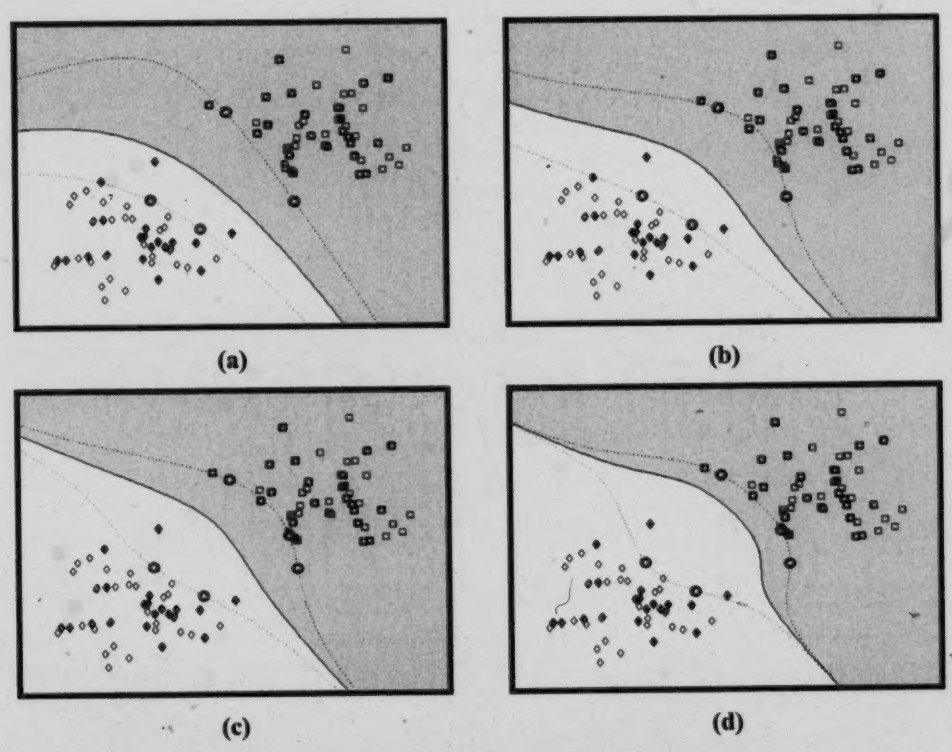


Figure 4-10: Decision boundaries generated by the SVM with the polynomial kernel (C=100) for d=2 (a), d=3 (b), d=5 (c), and d=10 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern.

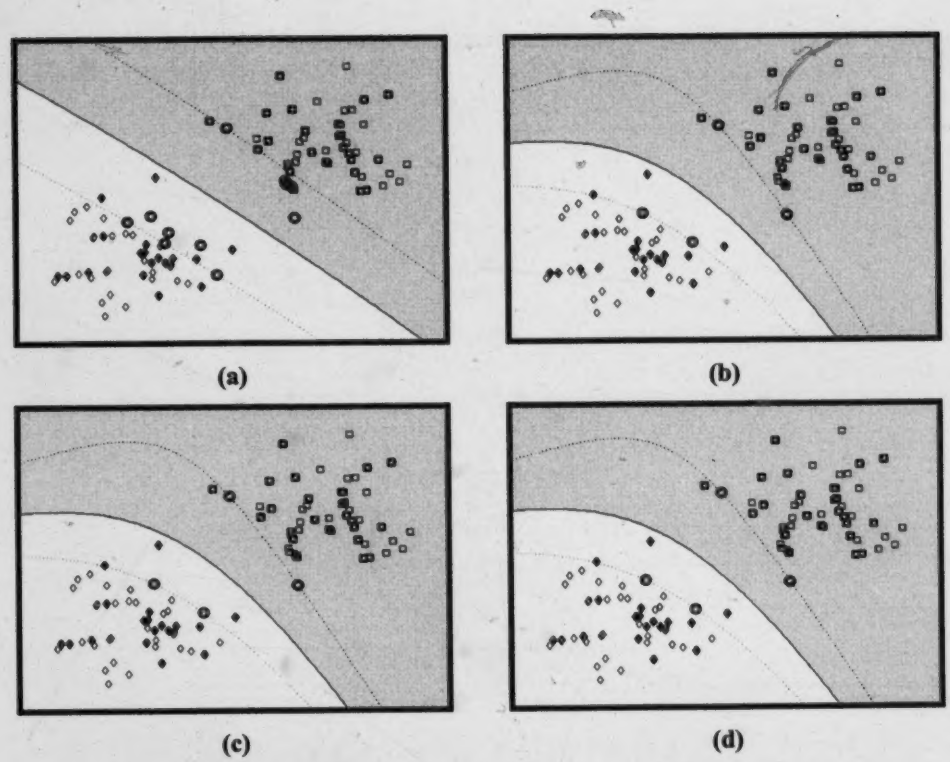


Figure 4-11: Decision boundaries generated by the SVM with the polynomial kernel (d=2) for C=0.1 (a), C=1 (b), C=10 (c), and C=1000 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern.

For the radial basis kernel, the decision boundaries for C=100 and several values of γ are shown in Figure 4-12 (γ=0.005 in Figure 4-12 (a), γ=0.05 in Figure 4-12 (b), γ=0.5 in Figure 4-12 (c), and γ=5 in Figure 4-12 (d)). From this figure, it can be seen that the decision boundaries change when the value of γ is changed (see Figure 4-12 (a) and Figure 4-12 (d)). Moreover, the number of support vectors also changes when the value of γ is changed (see Figure 4-12 (c) and Figure 4-12 (d)).

For the radial basis function kernel, the decision boundaries for γ=0.1 and several values of C are shown in Figure 4-13 (C=0.1 in Figure 4-13 (a), C=1 in Figure 4-13 (b), C=10 in Figure 4-13 (c), and C=1000 in Figure 4-13 (d)). From this figure, it can be seen that the

decision boundaries with smaller C (see Figure 4-13 (a) and Figure 4-13 (b)) have a larger margin and a larger number of support vectors, while the solutions with larger C (see Figure 4-9 (c) and Figure 4-13 (d)) have a smaller number of support vectors. Moreover, there is not a dramatic difference between the decisions boundaries generated for C=10 and C=1000 (see Figure 4-13 (c) and Figure 4-13 (d)).

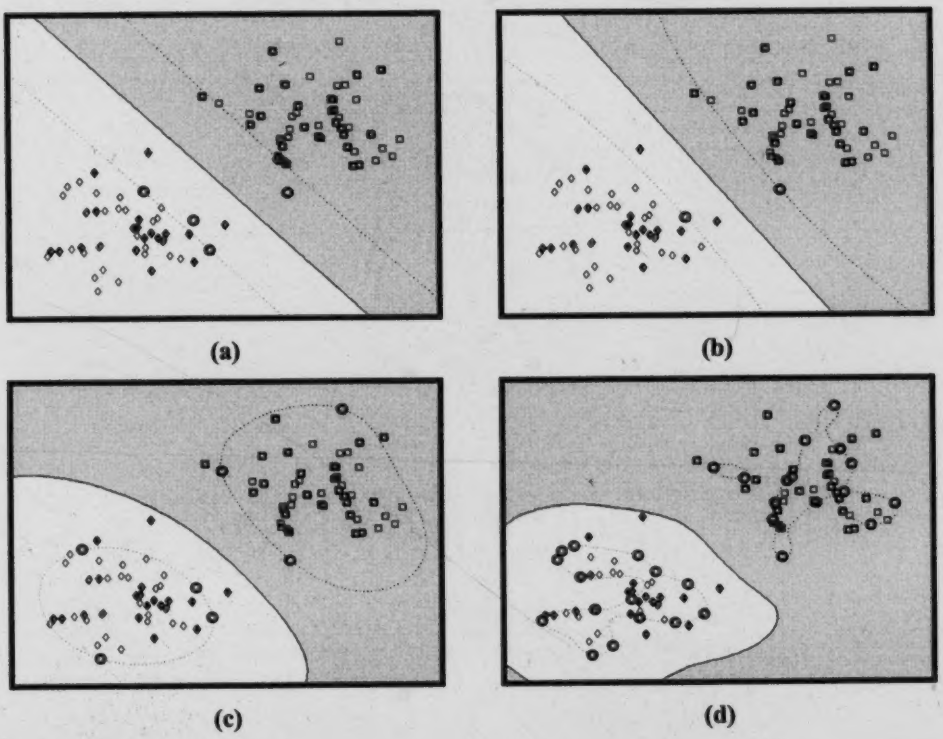


Figure 4-12: Decision boundaries generated by the SVM with the RBF kernel (C=100) for γ=0.005 (a), γ=0.05 (b), γ=0.5 (c), and γ=5 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern.

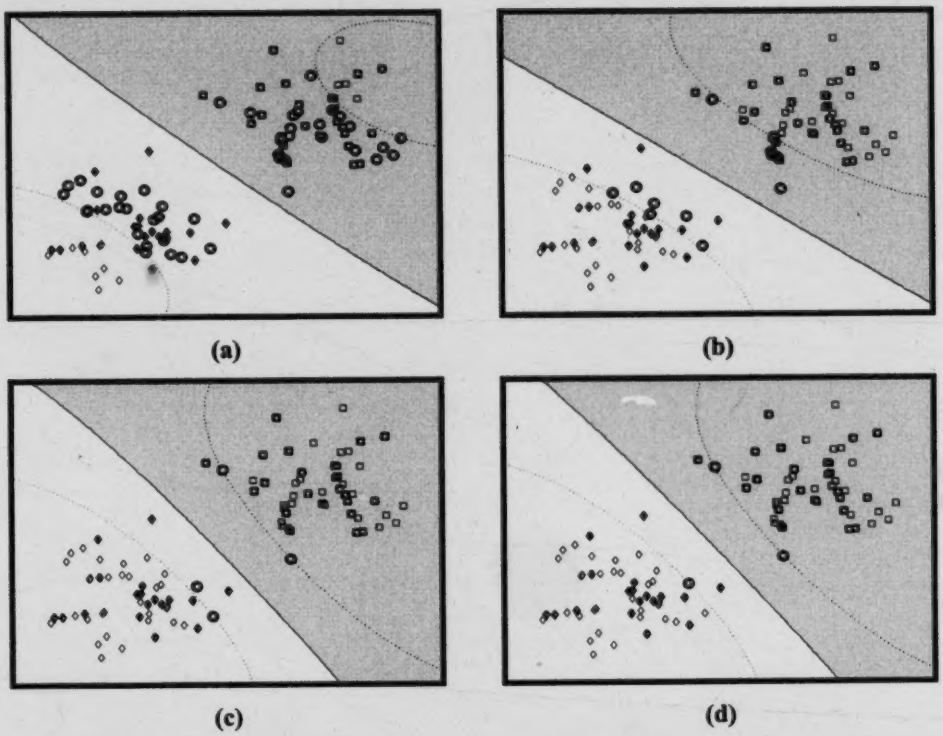


Figure 4-13: Decision boundaries generated by the SVM with the RBF kernel (γ=0.1) for C=0.1 (a), C=1 (b), C=10 (c), and C=1000 (d). The testing points are the bold patterns. *SV*'s are indicated with a circle around the corresponding pattern.

From Figure 4-9 through Figure 4-13, we conclude that the choice of the values of C and d or γ has a significant effect in the resulting shape and associated margin of the decision boundaries generated by the SVM classifier. Bearing these results in mind, in this thesis, a process of n_p-fold cross-validation is used to select the appropriate values for the kernel parameters. The process of cross-validation is performed in two stages. In the first one, cross-validation is done over a wide range of the parameters space to find an initial estimate of the parameters. In the second stage, the estimated parameters become more precise. This process is repeated n_p times. Finally, the kernel parameters are computed as the mean of the n_p previous estimations. The algorithm for selecting the values of the kernel parameters is depicted in Table 4-2.

Given	The data set $(\mathbf{x}_1, y_1),(\mathbf{x}_2, y_2),...,(\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \Re^p$, $y_i \in \{-1,+1\}$
Defined	n_p, the number of portions in which the data set will be randomly divided. $\mathbf{C}^i = \{C_1, C_2, ..., C_j, ..., C_l\}$, initial space for the error penalty term. $\boldsymbol{\gamma}^i = \{\gamma_1, \gamma_2, ..., \gamma_j, ..., \gamma_l\}$, initial space for the spread. $\mathbf{C}^f = \{\mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_j, ..., \mathbf{R}_l\}$, final space for the error penalty term with $\mathbf{R}_j = \{C_1^j, C_2^j, ..., C_m^j\}$ $\boldsymbol{\gamma}^f = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_j, ..., \mathbf{S}_l\}$, final space for the spread with $\mathbf{S}_j = \{\gamma_1^j, \gamma_2^j, ..., \gamma_m^j\}$
Processing	For each one of the n_p combinations of local training set ($(n_p$-1)/n_p of the training data)/local test set (1/n_p of the training data) For each value of error penalty term C, $C \in \mathbf{C}^i$ For each value of spread γ, $\gamma \in \boldsymbol{\gamma}^i$ • Determine the error on the local test set for the SVM trained with the local training set using a combination of C and γ. • Store the value of the local test error. Choose the values of C and γ for which the local test error is minimized. For each value of error penalty term C_2, $C_2 \in \mathbf{R}_j$, if $C = C_j$ For each value of spread γ_2, $\gamma_2 \in \mathbf{S}_j$, if $\gamma = \gamma_j$ • Determine the error on the local test set for the SVM trained with the local training set using a combination of C_2 and γ_2. • Store the value of the local test error. Choose the values of C_2 and γ_2 for which the local test error is minimized. Store the values of C_2 and γ_2. Calculate C_{out} and γ_{out} as the mean of the stored values of C_2 and γ_2 respectively.
Result	Error Penalty term C_{out} and spread γ_{out}.

Table 4-2: Algorithm for setting the parameters of the radial basis function kernel.

4.5 Multi-class SVM

Until now, this thesis has considered only the two-class classification problem. However, in real world problems, more than two classes are often involved. To properly deal with these problems, the original SVM approach must be extended to the multi-class case [71], [59], [75]. This work will be concerned mainly with these two approaches: (1) the one versus the rest (1-v-R) [71], which is the standard method for solving multi-class problems, and the decision-directed acyclic graph (DDAG), which was recently proposed by Platt et al. [59].

4.5.1 1-v-R

In this method, k binary classifiers are constructed. Each classifier is trained to separate one class, j, from the remaining classes. This approach involves all the training points: the training points of class j with positive labels and the other training points with negative labels. The output $f(\mathbf{x})$ is obtained by

$$f(\mathbf{x}) = \arg\left[\max_{j=1,\ldots,k} g_j(\mathbf{x})\right] \tag{4-17}$$

where $g_j(\mathbf{x})$ is

$$g_j(\mathbf{x}) = \sum_{i/\mathbf{x}_i \in SV} y_i \alpha_i^o K(\mathbf{x}_i, \mathbf{x}) + b_o, \quad j = 1,\ldots,k \tag{4-18}$$

with k being the number of classes.

4.5.2 DDAG

In this method, $k(k\text{-}1)/2$ binary classifiers are constructed. Each classifier is trained to separate one class from another, so in this approach, the only training points that are involved are those from the corresponding two classes. To obtain the output, a type of exclusion approach is used in the form of a directed acyclic graph. See Figure 4-14 for an example with four classes.

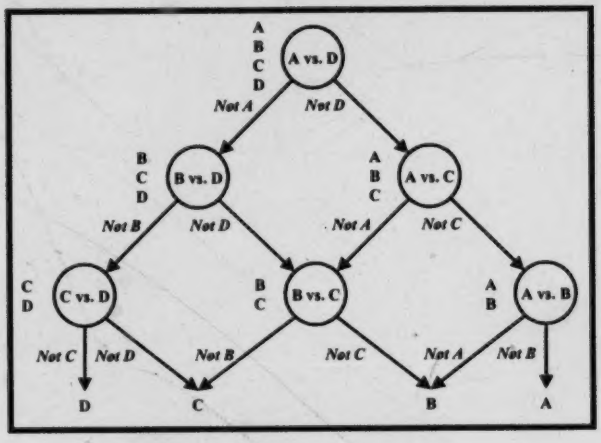


Figure 4-14: DDAG

An analysis of the training time required by the 1-v-R and the DDAG approaches to solve a multi-class classification problem is presented below.

Empirically, SVM training is observed to increase with the training set size according to the following expression [58]

$$T = an^b \tag{4-19}$$

where a is a proportionality constant, n is the number of training patterns, and b is usually equal to 2 when the SVM is implemented using sequential minimal optimization (see Section 4.6.4).

For the 1-v-R approach, the entire training set is used to train all k classifiers (with k being equal to the number of classes). Therefore, the training time for the 1-v-R approach is

$$T_{1\text{-v-R}} = kan^b \tag{4-20}$$

For the DDAG approach, only the training patterns of two classes are used to train $k(k\text{-}1)/2$ classifiers. Assuming that all the classes have the same number of elements, training one node of the DDAG requires $2n/k$ samples. Thus, the training time for the complete DDAG is

$$T_{\text{DDAG}} = \frac{k(k-1)}{2} a \left(\frac{2n}{k}\right)^b$$

(4-21)

From (4-20) and (4-21), the following expression can be derived

$$T_{\text{1-v-R}} = \frac{k^b}{(k-1)2^{b-1}} T_{\text{DDAG}}$$

(4-22)

From (4-22), we conclude that for increasing number of classes, the DDAG approach is faster than the 1-v-R approach in solving the multi-class classification problem.

Finally, even though these approaches have been successfully applied to solve multi-class classification problems, there is no a comprehensive study of their applicability, limitations, and associated error bounds.

4.6 Implementation techniques

This section briefly reviews some of the approaches used in training the SVM.

4.6.1 General Issues

The optimization problem associated with the SVM as classifier can be written as the maximization of

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

(4-23)

subject to

$$0 \le \alpha_i \le C$$

(4-24)

and

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

(4-25)

This problem is called quadratic programming (QP) because the function to be maximized depends quadratically on α, while α only appears linearly in the constraints.

The QP problem is solved when the Karush-Kuhn-Tucker (KKT) conditions are satisfied. This happens when for all i ($1 \le i \le n$, with n being the number of training patterns):

$$\alpha_i = 0 \Rightarrow y_i g(\mathbf{x}_i) \ge 1,$$
$$0 < \alpha_i < C \Rightarrow y_i g(\mathbf{x}_i) = 1,$$
$$\alpha_i = C \Rightarrow y_i g(\mathbf{x}_i) \le 1,$$

(4-26)

where α_i is the Lagrange multiplier associated with the ith input; y_i is the class label of the ith input; C is the error penalty term and $g(\mathbf{x})$ is given by

$$g(\mathbf{x}) = \sum_{i/\mathbf{x}_i \in SV} y_i \alpha_i^o K(\mathbf{x}_i, \mathbf{x}) + b_o$$

(4-27)

To solve the QP problem, one may use any of the optimization techniques developed over the years such as the Newton method, the conjugate gradient, the primal dual interior-point methods, etc. The main disadvantage of these methods is that many of them require the kernel matrix to be stored in memory. This matrix has a number of elements equal to the square of the number of training samples. Therefore, for large size problems a huge amount of memory may be required. For example, in the case of a data set of 30,000 training samples, storing the kernel matrix in memory requires 7,200 Mbytes (assuming that each element is stored as an 8-byte double precision number). Other approaches to solve the quadratic programming problem are described below.

4.6.2 The Gradient ascent approach

This method is described in [22]. The algorithm starts by assuming an initial solution, say α^o. Then it iteratively updates the vectors such that, at step τ, the vectors move a short distance in the direction of the greatest rate of increase of (4-23), i.e. in the direction of the gradient, evaluated at α^τ:

$$\Delta\alpha^\tau = \eta \nabla W \big|_{\alpha^\tau}$$

(4-28)

where the parameter η ($\eta > 0$) is the learning rate.

The drawbacks of this method are that it can be extremely slow in some data sets, a large value of η can cause the algorithm to oscillate without converging to the solution, and small values of η can result in a slow rate of convergence.

4.6.3 Chunking and Decomposition

The chunking method starts with an arbitrary subset 'chunk' of the data and trains the SVM with a standard QP solver. The support vectors are then retained, and all other data patterns in the chunk (with α_i=0) are discarded. Next, a new subset is formed with the support vector of the previous step and a subset of "*S*" of the remaining original patterns that most violate the KKT conditions when tested with the solution of the previous step for some value of *S*. This procedure is iterated until the stop condition is satisfied [22].

The chunking method assumes that the kernel matrix for the set of support vectors fits in memory so it can be fed to the QP solver being used. This assumption may cause the chunking algorithm to fail if the data set is too large or there are too many support vectors. To overcome this problem, Osuna et al. [47], [48] presented an algorithm (called decomposition) that uses a constant-size matrix, allowing for the training of arbitrarily large data sets. In this algorithm, the goal is to solve the QP problem by acting only on a small subset of the data at a time [22].

4.6.4 Sequential minimal optimization

Sequential minimal optimization (SMO) [58] is a method that can decompose the SVM QP problem without any extra matrix storage and without using standard QP solvers. In this method, only two α_i's are optimized at each iteration. The two elements to be optimized are chosen based on two heuristics, and the solution is found analytically. The first heuristic consists of choosing a pattern $\mathbf{x}_1$ that violates the KKT condition. The second choice heuristic consist of choosing a pattern $\mathbf{x}_2$ in such a way that updating α_1 (the Lagrange multiplier associated with pattern $\mathbf{x}_1$) and α_2 (the Lagrange multiplier associated with pattern $\mathbf{x}_2$) causes a large increase in (4-23).

A final remark regarding the SMO algorithm is that even though it requires more iterations to converge than the previous methods, the small number of operations that each iteration requires make the SMO faster than the other methods presented.

4.7 Relationship of SVM to partitional clustering

Before finishing this chapter, it is important to present a summary of the comparisons between SVM and partitional clustering.

- Both approaches store the learned data structure in a matrix (a partition matrix in the case of clustering and a kernel matrix in the case of SVM).
- After learning, both approaches select an usually small number of patterns as representatives of the data structure (prototypes in the case of clustering and support vectors in the case of SVM).
- They differ in the following two aspects:
  1. SVM requires the knowledge of the class labels while clustering does not.
  2. The similarities measures, used to determine the data structure, are performed in different spaces (the input space in the case of clustering and feature space in the case of SVM).

4.8 Summary

This chapter has introduced the SVM approach, a binary classification tool that has been successfully applied in a number of classification problems. This chapter has shown how the binary classification problem is solved using the SVM and how the choice of the kernel function affects the type of decision surfaces generated. The extension to the multi-class classifier was also studied, and some implementation techniques were discussed.

The following chapter will present another computational intelligence technique used for pattern recognition, artificial neural networks (specifically the radial basis functions neural networks).

5 Radial basis functions networks in pattern recognition

5.1 Introduction

The neural network approach is one of the approaches most commonly used for pattern recognition [41], [67]. An artificial neural network is a computational structure that is composed of a number of simple processors connected through a set of links, which have some weights associated with them. The most common architecture of neural networks used for pattern recognition is the feed-forward network, which includes single-layer perceptron, multilayer perceptron (MLP), and radial basis function (RBF) networks. This chapter gives a brief overview of RBF networks. Further information can be found in [9], [37].

This chapter begins with a brief introduction to neural networks. RBF neural networks are then studied, including approaches to determine their centres. Next, the centres of the RBF neural network are found using the orthogonal least squares approach [19] or prototypes stability analysis and FCM. Finally, an empirical comparison of the SVM with the RBF kernel against the RBF approaches is presented.

5.2 Background

An artificial neural network is a computational structure that is composed of a number of simple processors (neurons) connected in the form of a weighted directed graph [38]. There are two types of networks architectures: feed-forward networks, in which graphs have no loops (see Figure 5-1 a), and recurrent/feedback networks, in which loops occur (see Figure 5-1 b) [37], [11]. Feed-forward networks, which include the single-layer perceptron, the multilayer perceptron (MLP) [43], and the RBF neural networks [9], are normally trained in supervised mode (i.e. they require the knowledge of the desired output). Recurrent neural networks, such as self-organizing maps [40], the Hopfield networks [43], and the ART models [16]-[18], are normally used in unsupervised mode (i.e. they do not require previous knowledge about the expected output).

The learning process in a neural network involves modification of the network architecture and the connection weights to correctly classify the given input patterns. This property allows them to learn complex non-linear input-output relationships [38].

Neural network models are popular techniques for pattern recognition because of their ability to learn from data and the availability of efficient learning algorithms [38].

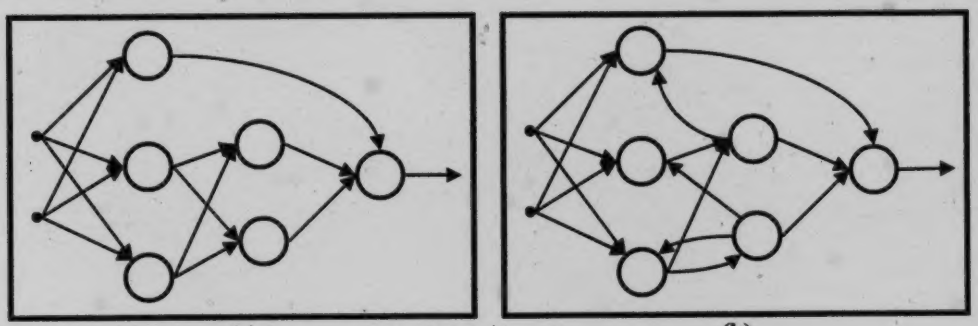


Figure 5-1: Feed-forward (a) and Feedback (b) neural network architectures.

5.3 Classic radial basis function (CRBF) neural networks

The RBF neural network [9] is a feed-forward neural network with two layers. The first layer consists of neurons normally equipped with Gaussian-like functions, even though other types of basis functions might be used, centred at the points specified by the weights associated with this layer. The second layer performs the linear combination of the outputs obtained from the first layer (see Figure 5-2).

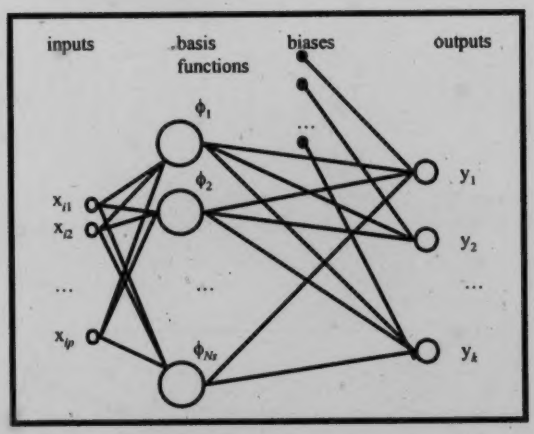


Figure 5-2: Architecture of a radial basis function neural network. $\mathbf{x}_i$ is the *i*th *p*-dimensional input pattern. *Ns* is the number of centres. *k* is the number of classes.

The decision boundary defined by the RBF classifier with Gaussian basis functions is given by:

$$f_j(\mathbf{x}) = sign\left(\sum_{i=1}^{Ns} w_{ji}\, exp\left(-\gamma_i \|\mathbf{x}-\mathbf{c}_i\|^2\right)+b_j\right), \gamma_i > 0, \quad j = 1,\ldots,k \tag{5-1}$$

where k represents the number of classes; Ns is the number of centres; $\mathbf{x}$ is an input pattern; w_{ji} are the weights for the second layer; γ_i represents the ith spread; b_j is the bias for the class "j"; and $\mathbf{c}_i$ is the ith centre vector used as a reference.

Considering a data set that falls into four classes as shown in Figure 5-3, one has that a multilayer perceptron separates the classes by forming hyperplanes in the input space (Figure 5-3 a), while an RBF classifier separates the classes by assigning a kernel to each class distribution separately (Figure 5-3 b).

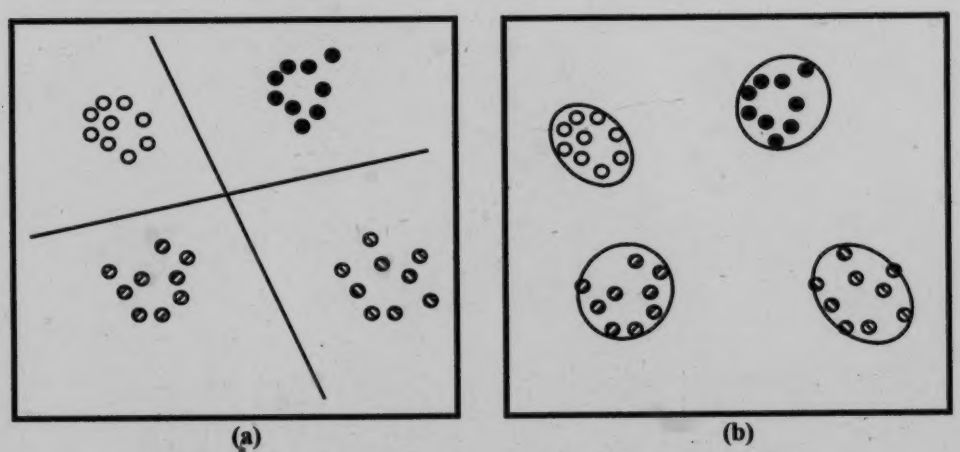


Figure 5-3: Schematic example of data points that fall into four distinct classes.

5.4 CRBF neural network training

For the training of a CRBF neural network, one may consider the two layers separately, leading to a two-stage training procedure. In the first stage, the centres $\mathbf{c}_i$ and γ_i are determined. In the second stage, the basis functions are kept fixed and the weights of the second layer are found.

The most commonly used approaches to find the centres are the orthogonal least squares (OLS) [19] and the application of clustering approaches, the latter of which includes

the k-means [36], the fuzzy c-means (FCM) [6], and self-organizing feature maps (SOM) [40]. In the OLS approach, centres (training points) are added until some preset performance is reached or all the training points have been used. In the clustering methods, the input data set is partitioned into several groups in such a way that the similarity among members of a group is larger than the similarity among groups. In this case, the centres represent the prototypes of each one of the resulting groups.

Depending on how the centres are found, some heuristics approaches set the values of γ_i as a function of the distance between the basis function layer's centres, while other heuristics set the values of γ_i as a function of the variance in each cluster. Other options to determine the values of γ_i involve using gradient-based methods or cross-validation.

Finally, the two basic approaches for finding the weights w_{ij} are the error backpropagation [66] and the pseudo-inverse method [60].

5.5 An empirical comparison of the SVM with the RBF against CRBF

The data sets for these experiments are depicted in Figure 5-4 and Figure 5-5. The vectors were distributed according to a normal density function with unit covariance matrices. The mean vectors (m_1 for class 1 and m_2 for class 2) were:

- For data (a): m_1=(2.0, 2.0) and m_2=(-2.0, -2.0). See Figure 5-4.
- For data (b): m_1=(1.0, 1.0) and m_2=(-1.0, -1.0). See Figure 5-5.

Each class had 50 elements, giving a data set of 100 elements, 60 of which were used for training and 40 for testing.

The experiments compare the performance of the SVM with the RBF kernel (SVM) against a CRBF classifier with the centres found using the OLS approach (CRBF1) and a CRBF classifier with the centres found using FCM (CRBF2).

For the SVM, the values of γ and C were determined using a two-stages process of cross-validation (on the training data). This process is explained in Table 5-1.

For the CRBF1, the performance criterion was set to reduce the mean square error during training up to a value G=5. The value γ was determined using a two-stages process of cross-validation (on the training data). This process is explained in Table 5-3.

For the CRBF2, prototypes stability analysis was performed to find the centres of the first layer. The second layer was trained to correctly classify the input data. The value of γ was

determined using a two-stages process of cross-validation (on the training data). This process is explained in Table 5-3.

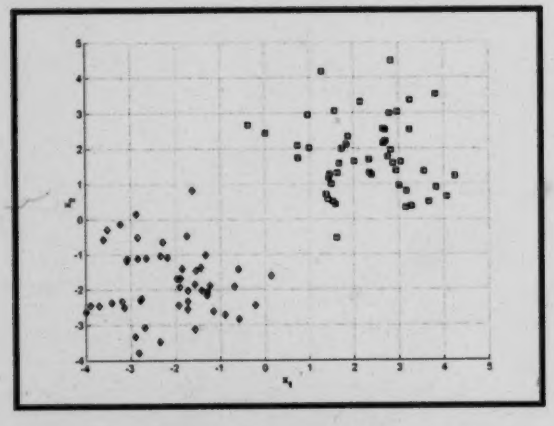


Figure 5-4: Synthetic data (a). Squares and diamonds represent class 1 and class 2, respectively.

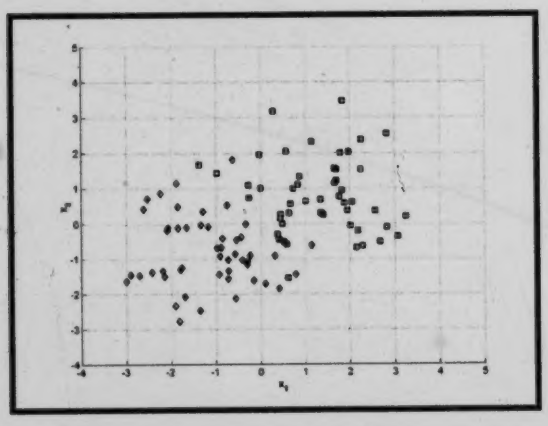


Figure 5-5: Synthetic data (b). Squares and diamonds represent class 1 and class 2, respectively.

To obtain reliable results, a rotation method was employed by randomly re-sampling training and testing data and repeating the experiments ten times. The results were averaged over the ten iterations.

Given	The data set $(\mathbf{x}_1, y_1),(\mathbf{x}_2, y_2),...,(\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \Re^p$, $y_i \in \{-1,+1\}$
Defined	n_p, the number of portions in which the data set will be randomly divided. $\mathbf{C}^i = \{C_1, C_2,..., C_j,..., C_l\}$, initial space for the error penalty term. $\boldsymbol{\gamma}^i = \{\gamma_1, \gamma_2,..., \gamma_j,..., \gamma_l\}$, initial space for the spread. $\mathbf{C}^f = \{\mathbf{R}_1, \mathbf{R}_2,..., \mathbf{R}_j,..., \mathbf{R}_l\}$, final space for the error penalty term with $\mathbf{R}_j = \{C_1^j, C_2^j,..., C_m^j\}$ $\boldsymbol{\gamma}^f = \{\mathbf{S}_1, \mathbf{S}_2,..., \mathbf{S}_j,..., \mathbf{S}_l\}$, final space for the spread with $\mathbf{S}_j = \{\gamma_1^j, \gamma_2^j,..., \gamma_m^j\}$
Processing	For each one of the n_p combinations of local training set ($(n_p-1)/n_p$ of the training data)/local test set ($1/n_p$ of the training data) For each value of error penalty term C, $C \in \mathbf{C}^i$ For each value of spread $\gamma, \gamma \in \boldsymbol{\gamma}^i$ • Determine the error on the local test set for the SVM trained with the local training set using a combination of C and γ. • Store the value of the local test error. Choose the values of C and γ for which the local test error is minimized. For each value of error penalty term C_2, $C_2 \in \mathbf{R}_j$, if $C = C_j$ For each value of spread γ_2, $\gamma_2 \in \mathbf{S}_j$, if $\gamma = \gamma_j$ • Determine the error on the local test set for the SVM trained with the local training set using a combination of C_2 and γ_2. • Store the value of the local test error. Choose the values of C_2 and γ_2 for which the local test error is minimized. Store the values of C_2 and γ_2. Calculate C_{out} and γ_{out} as the mean of the stored values of C_2 and γ_2 respectively.
Result	Error Penalty term C_{out} and spread γ_{out}.

Table 5-1: Algorithm for setting the parameters of the radial basis function kernel.

The set of parameters to be used in the process of cross-validation, in the case of the SVM, is shown in Table 5-2. This set was selected after gathering empirical evidence from experiments with different sets of parameters. This was achieved by monitoring the resulting number of support vectors generated by the SVM trained with each combination. It was noticed that sets with very small values for C and γ produce results with a high number of support vectors. These sets were discarded because as stated in [71], the expectation of the

probability of error for optimal hyperplanes is less than or equal to the expected number of support vectors divided by the number of training point plus one. Therefore, reducing the number of support vectors, the classifier may reduce the expectation of the probability of error.

$C \in \{100,1000\}$	$\gamma \in \{0.1,1\}$
$C_2 \in \{10,50,100,500\}$, if $C = 100$	$\gamma_2 \in \{0.05,0.1,0.2,0.5\}$, if $\gamma = 0.1$
$C_2 \in \{100,500,1000,5000\}$, if $C = 1000$	$\gamma_2 \in \{0.5,1,2,5\}$, if $\gamma = 1$

Table 5-2: Set of parameters to be used in the process of cross-validation, in the case of the SVM, for the artificial-data classification problem.

Given	The data set $(\mathbf{x}_1, y_1),(\mathbf{x}_2, y_2),...,(\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \Re^p$, $y_i \in \{-1,+1\}$
Defined	n_p, the number of portions in which the data set will be randomly divided. $\boldsymbol{\gamma}^i = \{\gamma_1, \gamma_2, ..., \gamma_j, ..., \gamma_l\}$, initial space for the spread. $\boldsymbol{\gamma}^f = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_j, ..., \mathbf{S}_l\}$, final space for the spread with $\mathbf{S}_j = \{\gamma_1^j, \gamma_2^j, ..., \gamma_m^j\}$
Processing	For each one of the n_p combinations of local training set ($(n_p$-1)/n_p of the training data)/local test set (1/n_p of the training data). For each value of spread $\gamma, \gamma \in \boldsymbol{\gamma}^i$ • Determine the error on the local test set for the CRBF trained with the local training set. • Store the value of the local test error. Choose the values of γ for which the local test error is minimised. For each value of spread γ_2, $\gamma_2 \in \mathbf{S}_j$, if $\gamma = \gamma_j$ • Determine the error on the local test set for the CRBF trained with the local training set. • Store the value of the local test error. Choose the values of γ_2 for which the local test error is minimised. Store the value of γ_2. Calculate γ_{out} as the mean of the stored values of γ_2.
Result	Spread γ_{out}.

Table 5-3: Algorithm for setting the spread of a radial basis function neural network.

The sets of parameters to be used in the process of cross-validation are shown in Table 5-4, in the case of the CRBF1, and in Table 5-5, in the case of the CRBF2. These sets were selected after gathering empirical evidence from experiments with different sets of parameters. This was achieved by monitoring the time required to train the classifiers. The sets that produced the best ratio of accuracy in training to time required to train the classifiers were selected.

$\gamma \in \{0.01, 0.1, 1\}$
$\gamma_2 \in \{0.001, 0.002, 0.005, 0.01\}$, if $\gamma = 0.01$
$\gamma_2 \in \{0.01, 0.02, 0.05, 0.1\}$, if $\gamma = 0.1$
$\gamma_2 \in \{0.1, 0.2, 0.5, 1\}$, if $\gamma = 1$

Table 5-4: Set of parameters to be used in the process of cross-validation, in the case of the CRBF1, for the artificial-data classification problem.

$\gamma \in \{0.001, 0.01, 0.1\}$
$\gamma_2 \in \{0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005\}$, if $\gamma = 0.001$
$\gamma_2 \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05\}$, if $\gamma = 0.01$
$\gamma_2 \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$, if $\gamma = 0.1$

Table 5-5: Set of parameters to be used in the process of cross-validation, in the case of the CRBF2, for the artificial-data classification problem.

The results for data (a) are summarized in Table 5-6 and the results for data (b) are summarized in Table 5-7. From these results, it can be seen that for data (a) the SVM and the CRBF2 had a perfect accuracy (100% in testing) while the performance of the CRBF1 was slightly lower (99.25% in testing). In the case of data (b), the SVM had the best accuracy in average (91.75% in testing), followed by the CRBF1 (with 91.5% in testing) and the CRBF2 (with 90.75% in testing).

Type of Classifier	Average classification accuracy (training)	Average classification accuracy (testing)	Average number of support vectors or centres
SVM	100%	100%	4.5
CRBF1	99.17%	99.25%	1.2
CRBF2	100%	100%	3.5

Table 5-6: Classification results obtained with different classifiers for data (a).

Type of Classifier	Average classification accuracy (training)	Average classification accuracy (testing)	Average number of support vectors or centres
SVM	96%	91.75%	11.1
CRBF1	96%	91.5%	7.1
CRBF2	95.67%	90.75%	3.7

Table 5-7: Classification results obtained with different kernels for data (b).

The graphical representation of the decision boundaries for the four classifiers on data (a) can be seen: in Figure 5-6 for the SVM (for γ=0.04 and C=10), in Figure 5-7 for the CRBF1 (for γ=0.04 and G=5), and in Figure 5-8 for the CRBF2 (for γ=0.04). From these figures, we conclude that:

- All the classifiers correctly classify the training and testing data.
- The SVM with RBF kernel maximizes the distance between the closest data point and the decision boundary.
- The CRBF1 and the CRBF2 generate a decision boundary that does a good job of separating the two classes but does not maximize the distance to the closest data point.

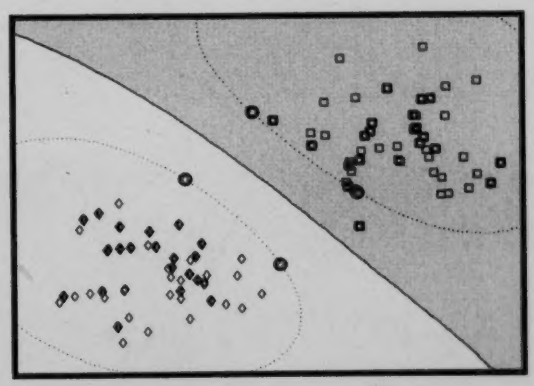


Figure 5-6: Decision boundaries for the SVM on data (a). Testing points are the bold patterns. Support vectors are indicated with a circle around the corresponding patterns.

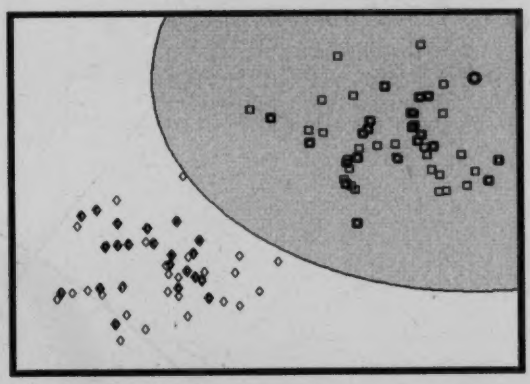


Figure 5-7: Decision boundaries for the CRBF1 on data (a). Testing points are the bold patterns. The centre is indicated with a circle around the corresponding pattern.

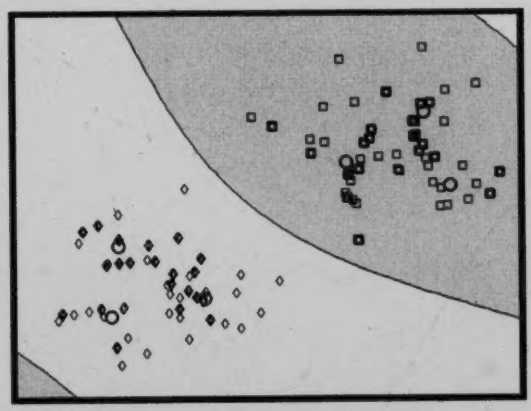


Figure 5-8: Decision boundaries for the CRBF2 on data (a). Testing points are the bold patterns. Centres are indicated with a circle.

Below is a summary of the comparisons between the SVM and the radial basis function neural networks.

- Both the SVM and the radial basis function neural networks learn from experimental data.
- After learning, both approaches are given the same mathematical model.

  For the radial basis function neural network, we have that

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{N_s} w_i \exp\left(-\gamma_i \|\mathbf{x}-\mathbf{c}_i\|^2\right)+b\right), \gamma_i > 0$$

For the SVM, we have that

$$f(x) = sign\left(\sum_{i/\mathbf{x}_i \in SV} y_i \alpha_i^o \, exp\left(-\gamma_i \|\mathbf{x} - \mathbf{x}_i\|^2\right) + b_o \right) \quad , \gamma_i > 0$$

- They differ by the learning method used. While RBF neural networks adjust their weights to reduce the error during training, the SVM's adjust their weights to maximize the margin in the training data.

5.6 Summary

This chapter has provided a brief overview of neural networks. The RBF approach was then described as well as some common approaches for its training. Finally, an empirical comparison of the RBF approaches studied so far (the SVM with the RBF kernel, the CRBF with the centres found using the OLS approach, and the CRBF with the centres found using FCM) for solving an artificial-data classification problem was presented. In the next chapter, fuzzy kernels are developed to incorporate fuzzy set methods into the SVM approach.

6 Designing Support Vector Machines with the Use of Fuzzy Granulation: Some Preliminary Results

6.1 Introduction

Even though support vector machines have a superb learning capability, it is almost impossible to come up with a reasonable interpretation of the results produced by them. Moreover, support vector machines were not developed to deal with problems stated in a linguistic way, situation that could often appears in practice of pattern recognition [50]. In contrast, fuzzy set technology hinge on linguistic terms [77], [79], [80], [56] to summarize the domain knowledge explicitly. Consequently, the results are clearly interpretable. Nevertheless, the learning ability of fuzzy systems is very limited. This chapter is concerned with the incorporation of fuzzy set methods into the support vector machine approach. In particular, fuzzy kernels are developed to allow for the manipulation of fuzzy information granules.

This chapter begins with a general discussion on merging fuzzy sets and support vector machines. The generation of the information granules is then presented. Next, fuzzy kernels are introduced, followed by the general architecture of the resulting approach. Finally, some numerical experiments are presented.

6.2 Fuzzy sets and support vector machines

This chapter considers using fuzzy sets in the design of support vector machines originating an approach that will be referred to as hybrid SVM (HSVM). The units of the HSVM model in which fuzzy sets appear are pre-processing, processing, and interpretation of the results (see Figure 6-1).

The pre-processing is performed by the fuzzy interface. It transforms the input data through the mapping $\Re^p \to [0,1]^{pc}$ where p is the dimensionality of the input data and c is the number of information granules. The resulting representation is appropriate for the HSVM operation because the HSVM operates at the level of information granules (more exactly, their levels of activation caused by the input data). The main roles of the input interface are related to non-linear data normalization (because the input data is normalized into [0,1] interval through a non-linear mapping that is driven by information granules defined in the input

space), variable processing resolution (because information granules are capable of focusing on suitable regions of the input variables [53] and capable of covering symbolic and numeric data [34]), and uncertainty representation (because fuzzy set theory provides a conceptual framework to represent and handle uncertainty in the input space [50]).

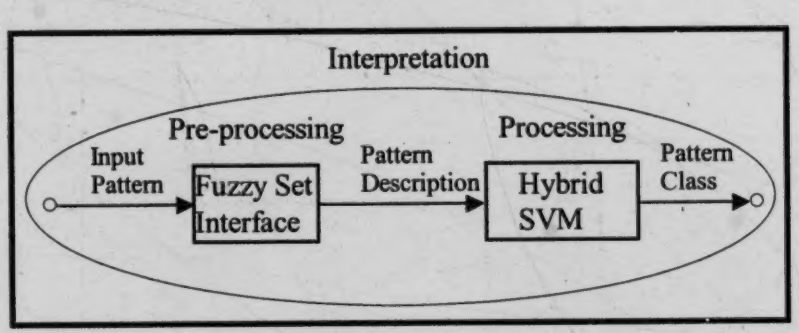


Figure 6-1: The scheme of pattern recognition using hybrid support vector machines (HSVM).

Processing involves handling the information coming from the fuzzy set interface. To do so, the system must incorporate fuzzy kernels. These fuzzy kernels allow for computation of similarities in the fuzzy sets space.

Interpretation of the results derives from analysis of the results and derivation of rules that represent the knowledge acquired by the hybrid support vector machine.

6.3 Generation of information granules

In this thesis, FCM is used to generate information granules. This is done in the following way: Prototypes stability analysis (see Section 3.4) is used to determine the appropriate number of information granules c. Fuzzy c-means is then applied to find c prototypes ($\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c$). Finally, the activation level of each information granules caused by a pattern $\mathbf{x}$ is given by

$$A_i(\mathbf{x}) = \frac{1}{\sum_{j=1}^{c} \frac{\|\mathbf{x}-\mathbf{v}_i\|^{2/(m-1)}}{\|\mathbf{x}-\mathbf{v}_j\|^{2/(m-1)}}} \quad , \; 1 \le i \le c \qquad (6\text{-}1)$$

where m is the fuzzification factor (see Section 3.3) and $\|.\|$ stands for a distance function defined between **x** and the respective prototype (see Section 3.3).

6.4 Fuzzy kernels

Kernels can be seen as similarity measures for the data in feature space. In the case of the hybrid support vector machine, the feature space consists of fuzzy sets. For this reason, the kernels (or fuzzy kernels, one may say) must be able to perform calculations that follow the rules of operations on fuzzy sets. The derivation of this type of kernels is presented in the next section.

6.4.1 Derivation

Matching neurons, a special case of referential logic-based neurons [52], [53], [55] are the logic structure behind the fuzzy kernels. In these neurons, the input signals are compared with respect to a given reference point. The results of this analysis are afterward aggregated using an OR neuron (disjunctive form of aggregation) or an AND neuron (conjunctive form of aggregation). Figure 6-2 and Figure 6-3 show a fuzzy kernel for a disjunctive form of aggregation and for a conjunctive form of aggregation respectively.

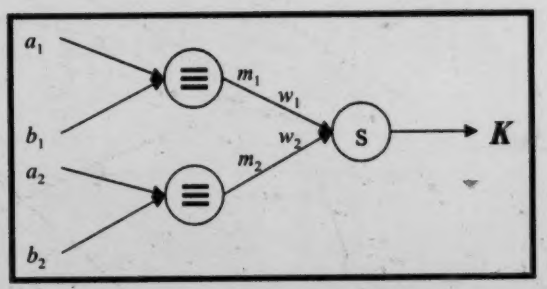


Figure 6-2: Fuzzy kernel for fuzzy inputs. The aggregation is disjunctive, m_1 and m_2 are the outputs of the matching neurons, and w_1 and w_2 are the aggregative neuron's connections.

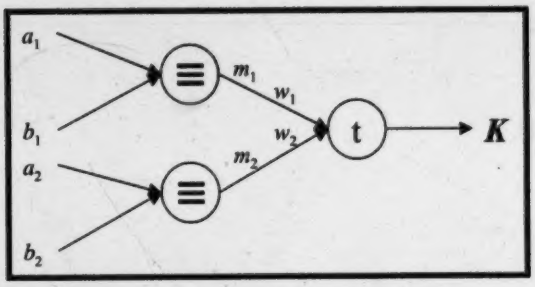


Figure 6-3: Fuzzy kernel for fuzzy inputs. The aggregation is conjunctive, m_1 and m_2 are the outputs of the matching neurons, and w_1 and w_2 are the aggregative neuron's connections.

The description of a fuzzy kernel for a disjunctive form of aggregation is

$$K_d = ((a_1 \equiv b_1) s\, w_1) t ((a_2 \equiv b_2) s\, w_2)$$

(6-2)

and for a conjunctive form of aggregation is

$$K_c = ((a_1 \equiv b_1) t\, w_1) s ((a_2 \equiv b_2) t\, w_2)$$

(6-3)

where w_1 and w_2 are the connections of the aggregative neuron, s and t are triangular norms [55], and the matching operation (≡) [51], [55] is given by

$$a \equiv b = 0.5\{(a \to b) \wedge (b \to a) + [(1-a) \to (1-b)] \wedge [(1-b) \to (1-a)]\}$$

(6-4)

with the implication (→) being defined as

$$(a \to b) = \sup\{c \in [0,1] / a\, t\, c \le b\}$$

(6-5)

The triangular norms used in this thesis were the algebraic product and probabilistic sum (i.e. $a t b = ab$ and $a s b = a + b - ab$), even though different norms may be used. For the product/sum norms, the matching operation is equal to

$$(a \equiv b) = \begin{cases} \frac{1}{2}\left(\frac{a}{b} + \frac{1-b}{1-a}\right), \text{if } a < b \\ \frac{1}{2}\left(\frac{b}{a} + \frac{1-a}{1-b}\right), \text{if } a > b \\ 1 \qquad\qquad\quad , \text{if } a = b \end{cases}$$

(6-6)

Figure 6-4 shows a fuzzy kernel (including the fuzzy set interface) for *p*-dimensional input data, *c* information granules, and a disjunctive form of aggregation.

Figure 6-5 shows the dimensionality of the data in each one of the stages of a fuzzy kernel.

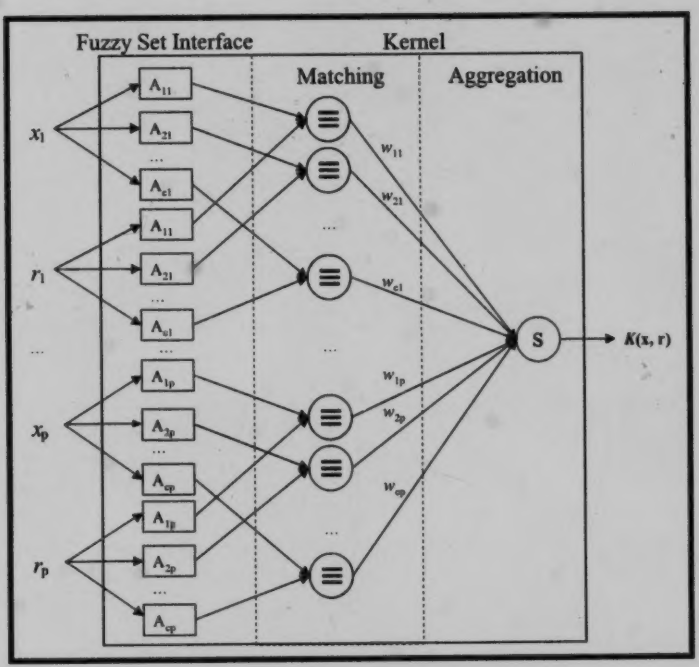


Figure 6-4: Fuzzy kernel as a superposition of matching and aggregative computation. In this case, the aggregation is disjunctive.

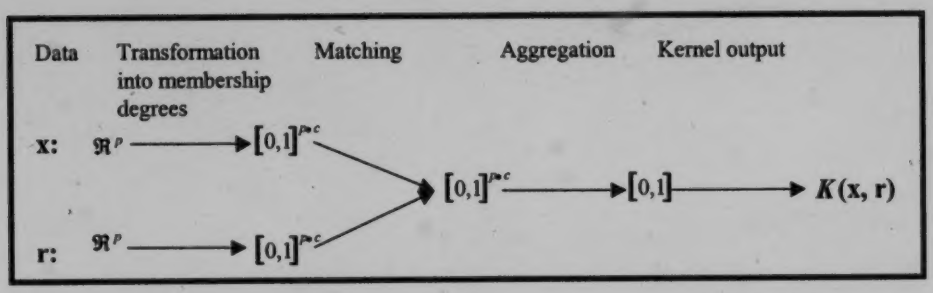


Figure 6-5: Mapping from a $\Re^p$ space of the input data to the [0,1] space of the kernel output, where p is the dimensionality of the input data and c is the number of information granules.

6.4.2 Fuzzy kernel outputs

These examples aim to illustrate the form of the outputs produced by the fuzzy kernels. For these experiments, the data represents activation levels of information granules allowing for a suitable graphical visualization. The input data consist of a group of vectors that were distributed uniformly in the [0, 1] interval and a reference vector that was set to $\mathbf{r}=[0.4\ 0.5]^T$. The connections of the aggregative neuron were set to $w_1=0.3$ and $w_2=0.3$. The characteristics for a conjunctive form of aggregation are depicted in Figure 6-6. The characteristics for a disjunctive form of aggregation are depicted in Figure 6-7.

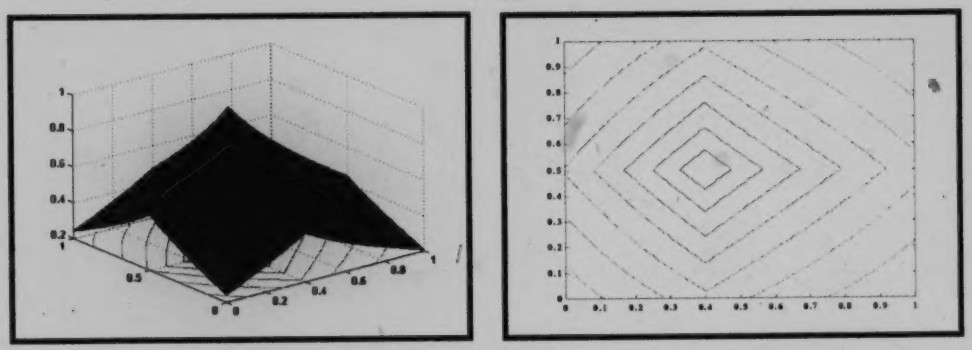


Figure 6-6: Non-linear characteristic of the kernel with conjunctive aggregation, with $w_1=0.3$, $w_2=0.3$, $b_1=0.4$, $b_2=0.5$, $a_1\in[0,1]$, $a_2\in[0,1]$, and the algebraic product and probabilistic sum norms.

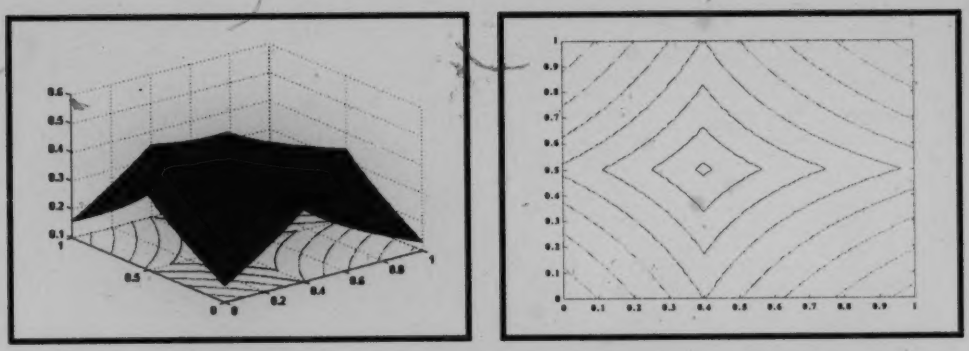


Figure 6-7: Non-linear characteristic of the kernel with disjunctive aggregation, with w_1=0.3, w_2=0.3, b_1=0.4, b_2=0.5, $a_1 \in [0,1]$, $a_2 \in [0,1]$, and the algebraic product and probabilistic sum norms.

6.4.3 Fuzzy kernels optimization

Fuzzy kernels weights will be optimized with the aid of genetic algorithms. In this section, background information related to genetic algorithms will be presented. Following that, the algorithm for optimizing the weights of the kernel and the error penalty term associated with the SVM approach will be described.

6.4.3.1 Background

Genetic algorithms [35], [29], [46] (GA's) are search techniques based on the principles of natural selection and evolution theory. A genetic algorithm (GA) operates on a population of chromosomes (or strings), which are basic units composed of a list of features called genes. Each chromosome is encoded to represent a solution to a given optimization problem. All chromosomes have an assigned fitness value that indicates how good their proposed solutions are. Before running a GA, the following prerequisites must be satisfied: the representation scheme must be chosen, the fitness measure formulated, the operations devised, the control parameters adjusted, and the termination condition determined.

The representation scheme can be seen as the encoding of the problem in a group of chromosomes. These artificial chromosomes can be strings of 1's and 0's, real numbers, parameter lists, computer programs, etc. The key issues in the representation scheme are the chromosomes' lengths and the alphabet symbols.

The fitness of a chromosome is a function $\boldsymbol{fit}: \Re^m \rightarrow \Re$ (with "m" being the dimensionality of the space of solutions), which is proportional to the goodness of the solution

represented by the chromosome. The fitness function makes it possible to discriminate a good solution from a bad one.

The main operators employed in the manipulation of chromosomes are reproduction, crossover, and mutation. Reproduction is a process in which individual strings are copied to a new population according to their fitness values. This can be accomplished in several ways. A weighted roulette wheel can be spun [29], a local tournament can be held [30], or a ranking procedure can be applied, etc. These methods can be combined with an elitist strategy, which implies passing a number of the fittest individuals to the next generation to avoid a decline in the fitness from one generation to the next. Crossover is an operator in which two parent chromosomes are combined to produce two new chromosomes. The crossover can be single-point, two-point, multiple points, or uniform, depending on how many portions are exchanged. Mutation is a process that is used to find new points in the search space by randomly altering a gene with small probability. It represents a random walk through the search space.

The control parameters of a GA are the population size, the number of generations to be run, the probability of crossover (p_c), and the probability of mutation (p_m). These parameters are usually tuned by hand, even though there have been several attempts to tune them automatically [25], [3], [44].

A termination criterion is set in such a way that the process is stopped when a given number of generations has been exceeded or a desired level of fitness function is achieved.

Once the steps for setting up a GA are completed, it can be run by following these steps (see Figure 6-8):

1. The initial population is obtained by generating, usually at random, a certain number of individuals.
2. The solutions are evaluated.
3. The genetic operators (reproduction, crossover, and mutation) are applied.
4. The solutions are evaluated.
5. The termination condition is verified. If it is not satisfied, then go to step 3.

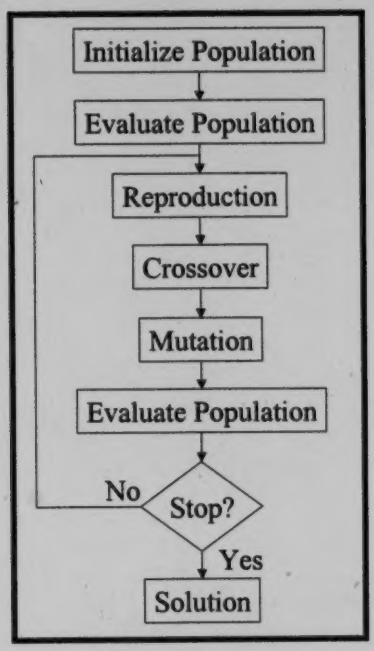


Figure 6-8: Major steps in a genetic algorithm.

Finally, the following is a list of features that separate GA's from most optimization algorithms [29]:

1. GA's operate with a coding of the input variables, not with the input variables themselves.
2. GA's search through a population of possible solutions, not a single solution.
3. GA's use fitness values only (no other information is needed).
4. GA's use probabilistic transition rules, not deterministic ones.

6.4.3.2 Optimization algorithm

Before presenting the optimization algorithm, the following aspects are described: the representation scheme, the fitness measure, the selection procedure, the reproduction operations, the adjustment of the control parameters, and the termination condition.

a. Representation: The optimization problem is encoded using real numbers that represent the strength of the connections (weights) and the error penalty term (C) associated with the SVM algorithm. See Figure 6-9. Real-coded GA's were selected because, compared to binary coded GA's, they have the following strengths [76]:

(1) increased efficiency because there is no need to convert bit strings into real numbers and (2) increased precision because there is no loss of precision due to binary representation.

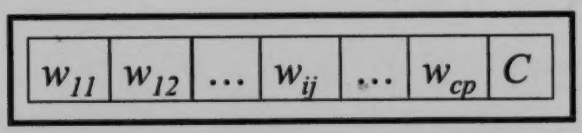


Figure 6-9: Fuzzy kernel parameters encoding. *C* is the error penalty term, w_{ij} is the weight of the connection *ij*, *c* is the number of information granules, *p* is the number of dimensions of the input data.

b. Fitness measure: the fitness function is given by

$$fit = (n - nsv)/(n - 2) \tag{6-7}$$

where n is the number of training patterns and nsv is the number of support vectors. The main objective of this fitness function is to promote a reduced number of support vectors. As stated in [71], the expectation of the probability of error for optimal hyperplanes is less than or equal to the expected number of support vectors divided by the number of training point plus one. Therefore, reducing the number of support vectors, the classifier may reduce the expectation of the probability of error.

c. Selection: First, select the two chromosomes with the best fitness and pass them to the next generation (elitism). Next, select a single individual by choosing two chromosomes randomly from the population and selecting the chromosome with the best fitness to survive to the next generation (binary tournament). The process is repeated PS-2 times (with PS being the population size).

d. Reproduction operators: the crossover and mutation operators used in this thesis are simple crossover [33]and uniform mutation [76]. The crossover operation is performed as follows: given $\mathbf{x}_1 = [x_1^1 \ \ldots \ x_j^1 \ \ldots \ x_m^1]$ and $\mathbf{x}_2 = [x_1^2 \ \ldots \ x_j^2 \ \ldots \ x_m^2]$ two chromosomes that have been selected to apply the crossover operator to them, a position $j \in \{1,2,\ldots,m-1\}$ is randomly chosen and two new chromosomes are built

$\mathbf{z}_1 = \left[x_1^1 \quad \ldots \quad x_j^1 \quad x_{j+1}^2 \quad \ldots \quad x_m^2\right]$ and $\mathbf{z}_2 = \left[x_1^2 \quad \ldots \quad x_j^2 \quad x_{j+1}^1 \quad \ldots \quad x_m^1\right]$. The mutation operation is performed as follows: given $\mathbf{x}_1 = \left[x_1^1 \quad \ldots \quad x_j^1 \quad \ldots \quad x_m^1\right]$ a chromosome that has been selected to apply the mutation operator to it, a randomly selected element x_j^1 with $x_j^1 \in \left[a_j, b_j\right]$, $j \in \{1,2,\ldots,m\}$ is replaced by z_j^1, which is a random number in the range $\left[a_j, b_j\right]$. The resulting chromosome is $\mathbf{z}_1 = \left[x_1^1 \quad \ldots \quad z_j^1 \quad \ldots \quad x_m^1\right]$.

e. Control parameters: selecting parameter setting for genetic algorithms, such as population size, crossover rate, and mutation rate, is often done by hand, even though there have been several works for setting them automatically. For example, Herrera and Lozano [32] introduced a rule-based approach to adjust the values of crossover and mutation rates. The rules have the following form:

if generation and population size then p_c (or p_m)

where p_c and p_m represent the crossover and mutation rates respectively. The complete collection of rules can be obtained from Table 6-1 in the case of p_c and in Table 6-2 in the case of p_m.

Regarding the last control parameter, population size, it is important to mention that if the population size is too large, the GA tends to take longer to converge upon a solution. However, if the population size is too small, the GA is in danger of premature convergence upon a suboptimal solution. This is primarily because there may not be enough diversity in the population to allow the GA to escape local optima. In general, the more difficult the problem is, the larger the population size should be. In practice, users typically perform empirical tests to set the value of the population size. They start with a small population size (say 8 to 20 chromosomes) and then they might try larger sizes (say 30 to 100 –or even more- chromosomes). Finally, they choose the population size that leaded to the best performance.

	Population Size		
Generations	Small	Medium	Big
Short	Medium	Small	Small
Medium	Large	Very Large	Medium
Long	Very Large	Very Large	Large

Crossover rate

Table 6-1: Rule bases for the control of p_c.

	Population Size		
Generations	Small	Medium	Big
Short	Large	Medium	Small
Medium	Medium	Small	Very Small
Long	Small	Very Small	Very Small

Mutation rate

Table 6-2: Rule bases for the control of p_m.

The genetic algorithm to optimize the fuzzy kernel parameters is the following:

1. Initialization. Initialize the values of weights (w_{ij}) with random numbers in [0, 1] and the value of C with a random number in [0, C_{max}] (with C_{max} being the maximum value C can take).
2. Evaluation. Apply the HSVM algorithm and calculate the fitness function which is given by (6-7).
3. Selection. Apply elitism to pass the best two chromosomes to the new generation. Apply local tournaments (of order two) to select the parents for the mating pool.
4. Crossover. Apply simple crossover to the chromosomes in the mating pool.
5. Mutation. Mutate, with probability equal to the mutation rate, each gene of the genes generated in step 4. Mutation consists of generating new random values subject to $w_{ij} \in [0, 1]$, with $1 \leq i \leq c$ and $1 \leq k \leq p$, and $C \in [0, C_{max}]$. Where c is the number of clusters and p is the number of dimensions of the input data.
6. Evaluation. Repeat step 2.
7. Check for termination condition. If the maximum number of generation is reached then stop. Otherwise, go to step 3.

6.5 General Scheme

The general architecture of a hybrid support vector machine is depicted in Figure 6-10. If there are not fuzzy granules available, the input data is used to generate information granules (via fuzzy c-means). The activation levels of each information granule caused by the input data are then determined. Next, the similarities of these activation levels are computed by using a fuzzy kernel. After that, the SVM algorithm is used to perform classification and to determine the number of support vectors (see Figure 6-11). Finally, a genetic algorithm is used to generate weights of the fuzzy kernels that minimize the number of support vectors.

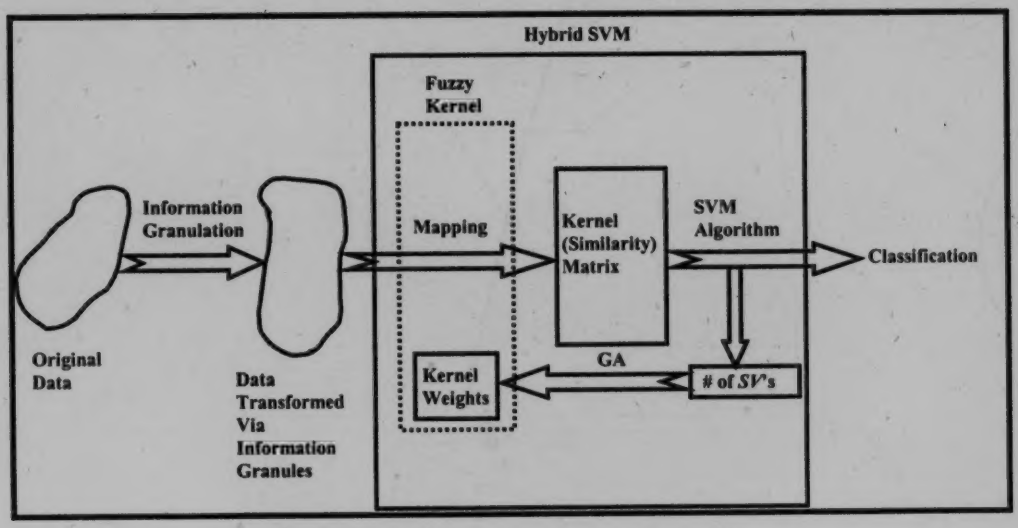


Figure 6-10: Hybrid Support Vector Machine.

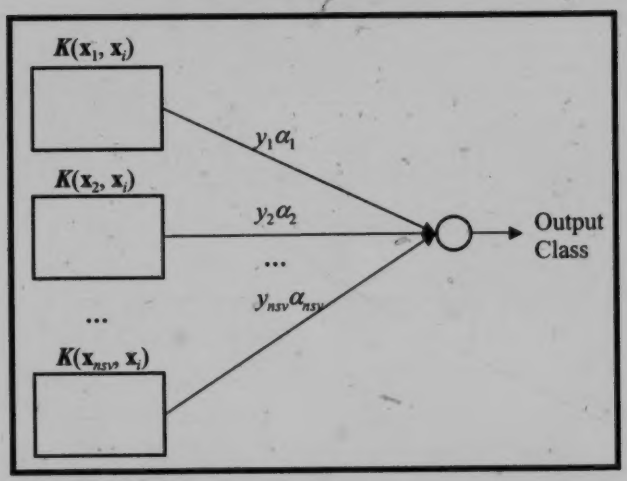


Figure 6-11: The SVM algorithm in the hybrid SVM has two layers. In the first layer, the similarity of the input data is computed by the fuzzy kernels $K(x_j, x_i)$ (for i=1,...,n and j=1,...,nsv with n being the number of input patterns and nsv being the number of support vectors). The second layer performs a linear combination of the outputs of the first layer. The weights of the second layer are given by the product of y_j (the label of the jth output variable in the training set) and α_j (the Lagrange multiplier, obtained by solving the quadratic programming problem associated with the SVM method, corresponding to the jth training vector).

The complete algorithm to apply the hybrid support vector machine for binary classification is given below (in the case of multi-class classification problems, the DDAG or 1-v-R approaches may be used)

Given	The data set $(\mathbf{x}_1, y_1),(\mathbf{x}_2, y_2),...,(\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \Re^p$, $y_i \in \{-1,+1\}$
Defined	*t*-norm and *s*-norm for the fuzzy kernel. Any parameter needed for the optimization algorithm.
Initialization	If required, use fuzzy c-means to generate the appropriate information granules.
Processing	• Transform the input data by finding their degree of membership (a value in the unit interval [0, 1]) in each information granule. • Use a genetic algorithm to determine the optimal values of the weights associated with the fuzzy kernels and the error penalty term associated with the SVM approach. • Train the hybrid support vector machine with the calculated error penalty term and weights.
Result	α (Lagrange multipliers). *SV*'s (support vectors). **w** (weights associated with the fuzzy kernels). *C* (error penalty term).

Table 6-3: Algorithm for implementing the HSVM approach.

6.6 Numerical studies

For these experiments, two classes of vectors were generated. The vectors were distributed according to a normal density function with unit covariance matrices. The mean vectors (m_1 for class 1and m_2 for class 2) were:

- For data (a): m_1=(2.0, 2.0) and m_2=(-2.0, -2.0). See Figure 6-12.
- For data (b): m_1=(1.0, 1.0) and m_2=(-1.0, -1.0). See Figure 6-13.

Each class had 50 elements, giving a data set of 100 elements, 60 of which were used for training and 40 for testing. To obtain reliable results, a rotation method was employed by randomly re-sampling training and testing data and repeating the experiments five times. The results were averaged over the five iterations. The experiments were conducted for the disjunctive aggregation kernel. The norms used were algebraic product and probabilistic sum.

The aggregative connections and the error penalty term were optimized by a genetic algorithm with the following parameters:

- Maximum number of generations: 30
- Population size: 50
- Crossover rate: 0.6
- Mutation rate: 0.01

This set of parameters was selected after gathering empirical evidence from experiments with different combinations of parameters. This was achieved by monitoring the number of support vectors and the accuracy over a validation set. The set that produced the best performance was selected.

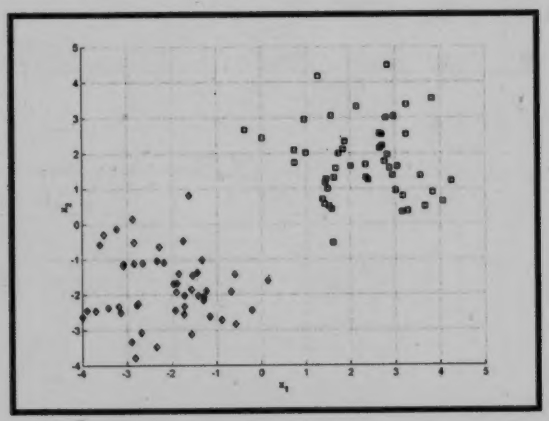


Figure 6-12: Synthetic data (a). Squares and diamonds represent class 1 and class 2 respectively.

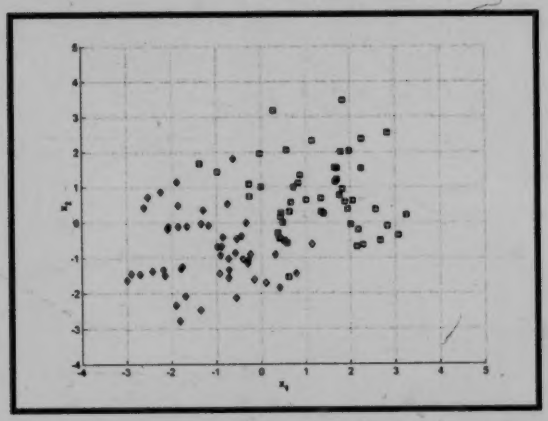


Figure 6-13: Synthetic data (b). Squares and diamonds represent class 1 and class 2 respectively.

The decision boundaries for one of the five experiments performed on data (a) are shown in Figure 6-14. From this figure, we conclude that more work must be done to produce decision surfaces with larger margin. This may lead to better generalization performances. The fitness function in successive generations, for the data displayed in Figure 6-14, is depicted in Figure 6-15. From this figure, it can be seen that the fitness of the best chromosome remained constant during all the generation.

The decision boundaries for one of the five experiments performed on data (b) are shown in Figure 6-16. From this figure, we conclude that, as indicated in the case of data (a), more work must be done to produce decision surfaces with larger margin. The fitness function in successive generations, for the data displayed in Figure 6-16, is depicted in Figure 6-17. From this figure, it can be seen that the fitness of the best chromosome remained constant after the seventh generation.

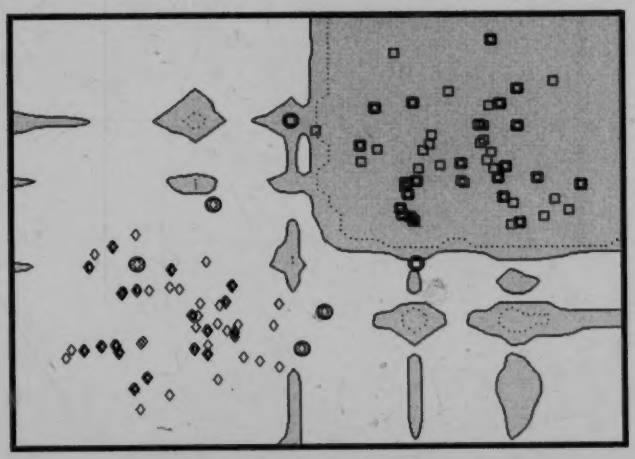


Figure 6-14: The discrimination boundaries produced by the HSVM classifier for data (a). Testing points are the bold patterns. Support vectors are indicated with a circle around the corresponding patterns.

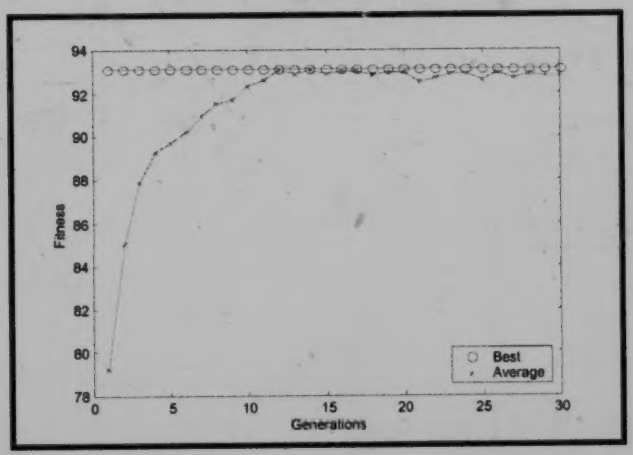


Figure 6-15: Fitness in successive generations for the data displayed in Figure 6-13.

Figure 6-16: The discrimination boundaries produced by the HSVM classifier for data (b). Testing points are the bold patterns. Support vectors are indicated with a circle around the corresponding training patterns. Errors in testing are indicated with an X across the testing pattern with the wrong shape.

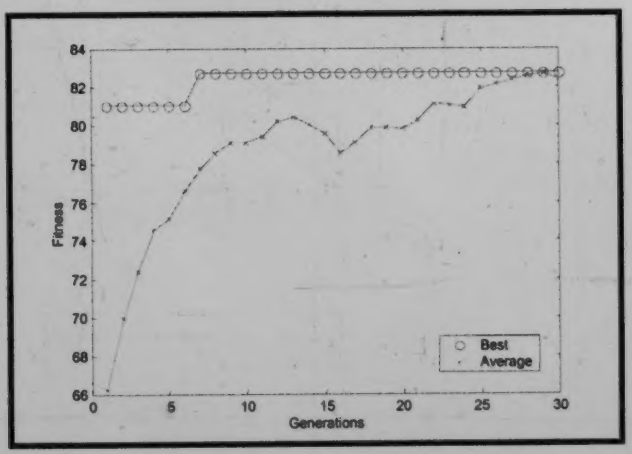


Figure 6-17:Fitness in successive generations for the data displayed in Figure 6-15.

The classification performance for both data sets is summarized in Table 6-4. From these results, we conclude that even though the number of support vectors was small, the performance over test data was not as good as expected. This may be related to the fact that the decision boundaries did not maximize the margin. Therefore, experiments involving a fitness function that takes into account the margin are required. Moreover, the number of

generations must be increased, and new reproduction operators, and control parameters must be tested to thoroughly study the performance of the HSVM approach.

Data set	Average classification accuracy (training)	Average classification accuracy (testing)	Average number of support vectors
Data (a)	98.33%	97%	4.4
Data (b)	90.67%	88.0%	12.8

Table 6-4: Classification results obtained with the HSVM.

6.7 Summary

This chapter has provided a brief introduction to the design methodology needed to allow for the incorporation of fuzzy set methods into the support vector machine approach. In this sense, fuzzy kernels were developed to provide the computational framework needed to calculate similarity in the fuzzy sets space. The generation of the information granules was also studied, and some numerical experiments with artificial data sets were presented and discussed.

7 Experimental Studies

7.1 Introduction

This chapter compares the performance of the methods described in the previous chapters (i.e. the RBF and the SVM approaches) with the performance of a 1-nearest neighbour classifier [21], [28]. These studies rely on widely available data sets, an iris data set, a Wisconsin breast cancer data set, and a Boston housing data, and a severe storm cell data set from Environment Canada.

The iris data set is available at the UCI Repository of Machine Learning[4]. It consists of patterns representing three varieties of irises: Setosa, Versicolour, and Virginica. The Wisconsin breast cancer data set is also available at UCI Repository of Machine Learning. It consists of records from 683 patients. Each record has nine attributes that are used to determine whether a tumour is benign or not. The Boston housing data set, from the UCI Repository of Machine Learning, is the third data set studied. It is concerned with house prices in the suburbs of Boston. The classification problem consists of determining whether a house price is greater than or equal to $21,000. The fourth data set, the severe storm cell data, is composed of meteorological volumetric data represented by a set of derived products. The classification problem is concerned with the classification of storm cells based on these derived features[5]. The problem has been divided into the following two types of sub-problems: (1) the binary classification problem, in which only hail and tornado storm cells are considered, and (2) the multi-class classification problem, in which four or ten categories of storm cells are considered. The criterion for comparison is the accuracy of the classification over a testing set.

7.2 Architectures

The experiments compare the performance of the SVM with the RBF kernel (SVM) against a CRBF classifier with the centres found using the OLS approach (CRBF1), a CRBF

---

[4] UCI Repository of Machine Learning Databases, C. Blake, E. Keogh, and C. J. Merz. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[5] Part of this work was done during my 2000 research assistantship at the University of Alberta under the supervision and guidance of Dr. W. Pedrycz and Dr. N. Pizzi. Without their detailed guidance, this work would not have been possible.

classifier with the centres found using FCM (CRBF2), and a k-nearest neighbour classifier [21], [28]. The HSVM was left aside because the decision boundaries it generated for the artificial data suggest that most work must be done to obtain simpler decision surfaces before using it to solve more complex classification problems.

For the SVM, the selected multi-class approaches were the 1-v-R approach and the DDAG approach, both of which used soft-margin classifiers. A two-stages process of cross-validation (on the training data) was used to determine the best combination for the values of γ and C. This process was explained in Table 5-1. The set of parameters to be used in the process of cross-validation is shown in Table 7-1. This set was selected after gathering empirical evidence from experiments with different sets of parameters. This was achieved by monitoring the resulting margin generated by the SVM trained with each combination. The set, which generated the largest margin, was the one selected.

$C \in \{1,10,100\}$	$\gamma \in \{0.001,0.01,0.1\}$
$C_2 \in \{0.1,0.5,1,5\}$, if $C = 1$	$\gamma_2 \in \{0.0001,0.0002,0.0005,0.001,0.002,0.005\}$, if $\gamma = 0.001$
$C_2 \in \{1,5,10,50\}$, if $C = 10$	$\gamma_2 \in \{0.001,0.002,0.005,0.01,0.02,0.05\}$, if $\gamma = 0.01$
$C_2 \in \{10,50,100,500\}$, if $C = 1000$	$\gamma_2 \in \{0.01,0.02,0.05,0.1,0.2,0.5\}$, if $\gamma = 0.1$

Table 7-1: Set of parameters to be used in the process of cross-validation, in the case of the SVM, for the real-data classification problems.

For the CRBF1, the performance criterion was set to reduce the mean square error during training up to a value G=5. The value γ was determined using a two-stages process of cross-validation (on the training data). This process was explained in Table 5-3. The set of parameters to be used in the process of cross-validation is shown in Table 7-2. This set was selected after gathering empirical evidence from experiments with different sets of parameters. This was achieved by monitoring the time required to train the classifiers. The set that produced the best ratio of accuracy in training to time required to train the classifiers was selected.

$\gamma \in \{0.01, 0.1, 1\}$
$\gamma_2 \in \{0.001, 0.002, 0.005, 0.01\}$, if $\gamma = 0.01$
$\gamma_2 \in \{0.01, 0.02, 0.05, 0.1\}$, if $\gamma = 0.1$
$\gamma_2 \in \{0.1, 0.2, 0.5, 1\}$, if $\gamma = 1$

Table 7-2: Set of parameters to be used in the process of cross-validation, in the case of the CRBF1, for the real-data classification problems.

For the CRBF2, prototypes stability analysis was performed to find the centres of the first layer. The second layer was trained to correctly classify the input data. The value of γ was determined using a two-stages process of cross-validation (on the training data). This process was explained in Table 5-3. The set of parameters to be used in the process of cross-validation is shown in Table 7-3. This set was selected after gathering empirical evidence from experiments with different sets of parameters. This was achieved by monitoring the time required to train the classifiers. The set that produced the best ratio of accuracy in training to time required to train the classifiers was selected.

$\gamma \in \{0.001, 0.01, 0.1\}$
$\gamma_2 \in \{0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005\}$, if $\gamma = 0.001$
$\gamma_2 \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05\}$, if $\gamma = 0.01$
$\gamma_2 \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$, if $\gamma = 0.1$

Table 7-3: Set of parameters to be used in the process of cross-validation, in the case of the CRBF2, for the real-data classification problems.

For the final classifier, the k-nearest neighbours approach was tested for $k \in \{1, 3, 5\}$ using Euclidean distance. The best result in terms of classification performance was given by k=1, i.e. the 1-nearest neighbour (1-NN) algorithm.

7.3 Iris Classification

The data set consists of 50 instances of each of the three species of iris: Setosa, Versicolour, and Virginica.

7.3.1 Methodology

In the experiments, 50% of the data set (i.e. 75 samples) was used as a training (learning) set and the remaining 50% (75 samples) was used as a testing set. To obtain reliable results, a rotation method was employed, repeating the experiments ten times. Subsequently, the training data for each one of the ten experiments were normalized to attain a zero mean and one standard deviation. The values for the mean and the standard deviation were then used to normalize the testing data for each experiment.

7.3.2 Comparative analysis

The experiments were performed independently over the 10 randomly re-sampled groups. The results described in this section include:

1. The average and standard deviation (in parenthesis) confusion matrices for the training and testing sets.
2. The average kappa score (κ), including its standard deviation (in parenthesis), for the training and testing sets.
3. The average classification error (E), including its standard deviation (in parenthesis), for the training and testing sets.

The labels for the confusion matrices are:

- To: True (desired) output
- Co: Classifier (actual) output
- S: Setosa
- Ve: Versicolour
- Vi: Virginica

The performance results are displayed in Table 7-4 through Table 7-8.

Training			
To\Co	S	Ve	Vi
S	**24.6(2.7)**	0(0)	0(0)
Ve	0(0)	**24.6(3.0)**	0.1(0.3)
Vi	0(0)	0.3(0.7)	**24.4(1.8)**

κ=0.97(0.01), E=1.87%(0.93%)

Testing			
To\Co	S	Ve	Vi
S	**25.4(2.7)**	0(0)	0(0)
Ve	0(0)	**24.4(2.6)**	0.9(0.9)
Vi	0(0)	3.1(1.3)	**21.2(2.2)**

κ=0.92(0.04), E=5.33%(2.43%)

Table 7-4: CRBF1 training and testing performance results for the iris data set.

Training			
To\Co	S	Ve	Vi
S	**24.6(2.7)**	0(0)	0(0)
Ve	0.1(0.3)	**22.1 (3.7)**	2.5(1.5)
Vi	0(0)	4.4(2.4)	**21.3(3.7)**

κ=0.86(0.07), E=9.33%(4.53%)

Testing			
To\Co	S	Ve	Vi
S	**25.4(2.7)**	0(0)	0(0)
Ve	0.9(1.3)	**21.2(2.5)**	3.2(3.9)
Vi	0(0)	4.4(2.1)	**19.9(2.4)**

κ=0.83(0.06), E=11.33%(3.78%)

Table 7-5: CRBF2 training and testing performance results for the iris data set.

Training			
To\Co	S	Ve	Vi
S	**24.6(2.7)**	0(0)	0(0)
Ve	0(0)	**24.7(3.0)**	0(0)
Vi	0(0)	0(0)	**25.7(1.6)**

κ=1(0), E=0%(0%)

Testing			
To\Co	S	Ve	Vi
S	**25.4(2.7)**	0(0)	0(0)
Ve	0(0)	**23(2.8)**	2.3(0.8)
Vi	0(0)	2.1(1.2)	**22.2(2.4)**

κ=0.91(0.03), E=5.87%(2.01%)

Table 7-6: 1-NN training and testing performance results for the iris data set.

Training			
To\Co	S	Ve	Vi
S	**22(8.2)**	2.6(7.9)	0(0)
Ve	0(0)	**23.5(2.8)**	1.2(0.9)
Vi	0(0)	0.9(1.3)	**24.8(2.3)**

κ=0.91(0.16), E=6.27%(10.52%)

Testing			
To\Co	S	Ve	Vi
S	**22.6(8.4)**	2.8(7.8)	0(0)
Ve	0(0)	**23.6(3.2)**	1.7(1.1)
Vi	0(0)	1.3(1.6)	**23(2.1)**

κ=0.88(0.16), E=7.73%(10.55%)

Table 7-7: DDAG training and testing performance results for the iris data set.

Training			
To\Co	S	Ve	Vi
S	**24.6(2.7)**	0(0)	0(0)
Ve	0(0)	**24(2.8)**	0.7(0.7)
Vi	0(0)	0.6(0.7)	**25.1(1.7)**

κ=0.97(0.02), E=1.73%(1.10%)

Testing			
To\Co	S	Ve	Vi
S	**25.4(2.7)**	0(0)	0(0)
Ve	0(0)	**23.6(3.1)**	1.7(1.0)
Vi	0(0)	1(1.3)	**23.3(2.0)**

κ=0.95(0.02), E=3.6%(1.55%)

Table 7-8: 1-v-R training and testing performance results for the iris data set.

The results averaged over ten iterations are summarized in Table 7-9, Figure 7-1, and Figure 7-2. The results include the accuracy in training and testing and the kappa score in training and testing. From these results, it can be seen that the SVM, in the 1-v-R approach, outperformed the CRBF1, the CRBF2, the 1-NN, and the DDAG classifiers showing the best accuracy (98.27% under testing) and the highest kappa score (0.95 under testing).

	CRBF1	CRBF2	1-NN	DDAG	1-v-R
Training Accuracy	98.13%	90.67%	100%	93.73%	98.27%
Testing Accuracy	94.67%	88.67%	94.13%	92.27%	96.4%
Training Kappa	0.97	0.86	1	0.91	0.97
Testing Kappa	0.92	0.83	0.91	0.88	0.95

Table 7-9: Summary of results for the iris data set.

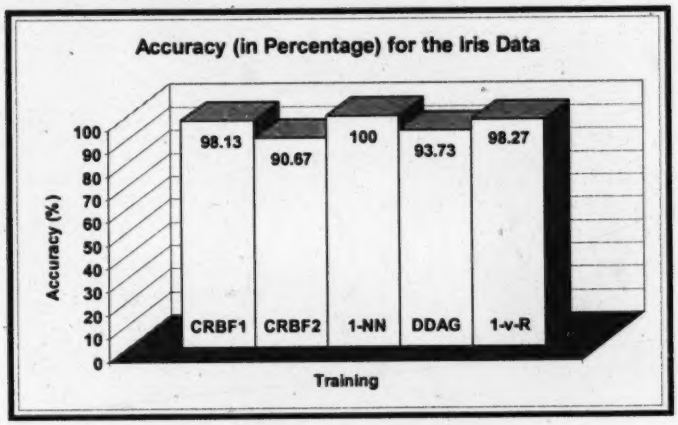


Figure 7-1: Training accuracy for the iris data set.

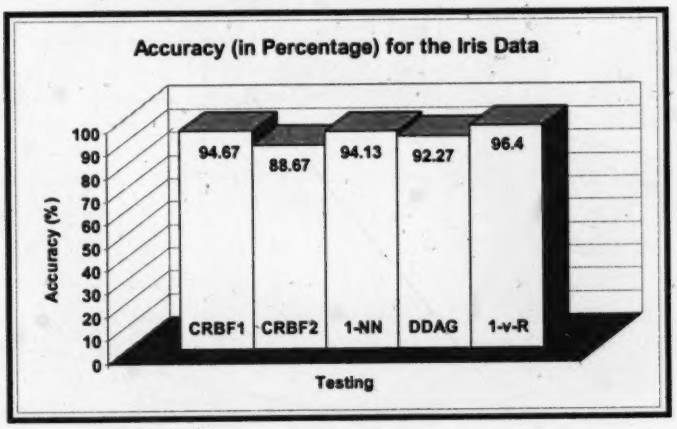


Figure 7-2: Testing accuracy for the iris data set.

7.4 Wisconsin Breast Cancer Classification

The data set consists of 683 records, of which 444 were confirmed to correspond to benign tumours and 239 of which were labelled as malign tumours.

7.4.1 Methodology

In the experiments, 60% of the data set (i.e. 409 records) was used as a training (learning) set and the remaining 40% (274 records) was used as a testing set. To obtain reliable results, a rotation method was employed, repeating the experiments ten times. Subsequently, the training data for each one of the ten experiments were normalized to attain a zero mean and one standard deviation. The values for the mean and the standard deviation were then used to normalize the testing data for each experiment.

7.4.2 Comparative analysis

The experiments were performed independently over the 10 randomly re-sampled groups. The results described in this section include:

1. The average and standard deviation (in parenthesis) confusion matrices for the training and testing sets.
2. The average kappa score (κ), including its standard deviation (in parenthesis), for the training and testing sets.
3. The average classification error (E), including its standard deviation (in parenthesis), for the training and testing sets.

The labels for the confusion matrices are:

- To: True (desired) output
- Co: Classifier (actual) output
- B: Benign
- M: Malign

The performance results are displayed in Table 7-10 through Table 7-13.

Training		
To\Co	B	M
B	**261.4(4.9)**	4.5(1.8)
M	3.1(1.4)	**140(6.7)**

κ=0.96(0.02), E=1.86%(0.71%)

Testing		
To\Co	B	M
B	**172.6(5.8)**	4.5(1.6)
M	5.6(3.2)	**90.3(7.8)**

κ=0.92(0.03), E=3.70%(1.19%)

Table 7-10: CRBF1 training and testing performance results for the Wisconsin breast cancer data.

Training		
To\Co	B	M
B	**259.6(5.8)**	6.3(1.6)
M	6.3(1.1)	**136.8(6.0)**

κ=0.93(0.01), E=3.08%(0.49%)

Testing		
To\Co	B	M
B	**173.7(5.8)**	3.4(1.6)
M	4.1(1.1)	**91.8(5.8)**

κ=0.94(0.02), E=2.75%(0.80%)

Table 7-11: CRBF2 training and testing performance results for the Wisconsin breast cancer data.

Training		
To\Co	B	M
B	**265.9(5.8)**	0(0)
M	0(0)	**143.1(5.8)**

κ=1(0), E=0%(0%)

Testing		
To\Co	B	M
B	**173.3(5.3)**	3.8(1.7)
M	6(3.0)	**89.9(5.7)**

κ=0.92(0.03), E=3.59%(1.12%)

Table 7-12: 1-NN training and testing performance results for the Wisconsin breast cancer data.

Training		
To\Co	B	M
B	**260.5(6.7)**	5.4(2.6)
M	3.5(2.1)	**139.6(6.4)**

κ=0.95(0.02), E=2.18%(0.88%)

Testing		
To\Co	B	M
B	**172.9(6.2)**	4.2(1.6)
M	4.6(2.4)	**91.3(5.9)**

κ=0.93(0.03), E=3.22%(1.24%)

Table 7-13: SVM training and testing performance results for the Wisconsin breast cancer data.

The results averaged over the ten iterations are summarized in Table 7-14, Figure 7-3, and Figure 7-4. The results include the accuracy in training and testing and the kappa score in training and testing. From these results, it can be seen that the CRBF2 followed by the SVM were the approaches that showed the best accuracy (97.25% and 96.78% respectively under testing) and the highest kappa score (0.94 and 0.93 respectively under testing).

	CRBF1	CRBF2	1-NN	SVM
Training Accuracy	98.14%	96.92%	100%	97.82%
Testing Accuracy	96.3%	97.25%	96.41%	96.78%
Training Kappa	0.96	0.93	1	0.95
Testing Kappa	0.92	0.94	0.92	0.93

Table 7-14: Summary of results for the Wisconsin breast cancer data set.

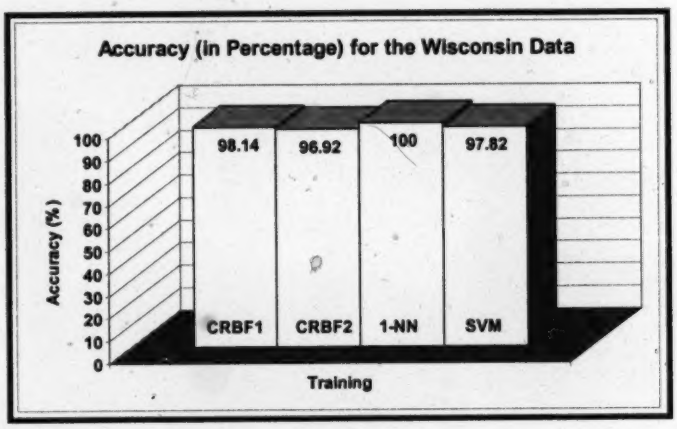


Figure 7-3: Training accuracy for the Wisconsin breast cancer data set.

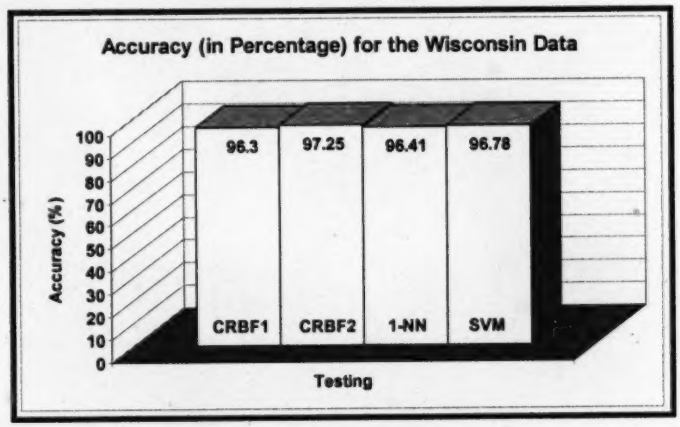


Figure 7-4: Testing accuracy for the Wisconsin breast cancer data set.

7.5 Boston Housing Classification

The data set consists of 506 instances, of which 260 correspond to house prices greater than or equal to $21,000 and 246 of which correspond to house prices less than $21,000.

7.5.1 Methodology

In the experiments, 60% of the data set (i.e. 304 patterns) was used as a training (learning) set and the remaining 40% (202 patterns) was used as a testing set. To obtain

reliable results, a rotation method was employed, repeating the experiments ten times. Subsequently, the training data for each one of the ten experiments were normalized to attain a zero mean and one standard deviation. The values for the mean and the standard deviation were then used to normalize the testing data for each experiment.

7.5.2 Comparative analysis

The experiments were performed independently over the 10 randomly re-sampled groups. The results described in this section include:

1. The average and standard deviation (in parenthesis) confusion matrices for the training and testing sets.
2. The average kappa score (κ), including its standard deviation (in parenthesis), for the training and testing sets.
3. The average classification error (E), including its standard deviation (in parenthesis), for the training and testing sets.

The labels for the confusion matrices are:

- To: True (desired) output
- Co: Classifier (actual) output
- A: House prices greater than or equal to $21,000
- B: House prices less than $21,000

The performance results are displayed in Table 7-15 through Table 7-18.

Training		
To\Co	A	B
A	**151.5(5.8)**	2.7(2.7)
B	2.6(2.6)	**146.2(6.0)**

κ=0.97(0.03), E=1.75%(1.61%)

Testing		
To\Co	A	B
A	**82.7(6.4)**	23.1(5.1)
B	19.1(7.3)	**78.1(6.8)**

κ=0.58(0.10), E=20.79%(5.19%)

Table 7-15: CRBF1 training and testing performance results for the Boston housing data.

Training		
To\Co	A	B
A	**134.7(7.2)**	19.5(4.9)
B	24.4(5.5)	**124.4(8.3)**

κ=0.71(0.06), E=14.49%(2.87%)

Testing		
To\Co	A	B
A	**91.1(5.3)**	14.7(3.7)
B	17.7(2.8)	**79.5(4.9)**

κ=0.68(0.04), E=15.96%(2.13%)

Table 7-16: CRBF2 training and testing performance results for the Boston housing data.

Training		
To\Co	A	B
A	**154.2(5.1)**	0(0)
B	0(0)	**148.8(5.1)**

κ=1(0), E=0%(0%)

Testing		
To\Co	A	B
A	**88.2(4.6)**	17.6(4.1)
B	14.2(3.8)	**83(4.2)**

κ=0.69(0.04), E=15.67%(2.2%)

Table 7-17: 1-NN training and testing performance results for the Boston housing data.

Training		
To\Co	A	B
A	**149.4(6.1)**	4.8(2.3)
B	4.3(3.0)	**144.5(6.0)**

κ=0.94(0.03), E=3.00%(1.32%)

Testing		
To\Co	A	B
A	**94(6.1)**	11.8(3.6)
B	13.2(4.3)	**84(5.1)**

κ=0.75(0.03), E=12.32%(1.58%)

Table 7-18: SVM training and testing performance results for the Boston housing data.

The results averaged over the ten iterations are summarized in Table 7-19, Figure 7-5, and Figure 7-6. The results include the accuracy and the kappa score in training and testing. From these results, it can be seen that the SVM was the approach that showed the best accuracy (87.68% under testing) and the highest kappa score (0.75 under testing).

	CRBF1	CRBF2	1-NN	SVM
Training Accuracy	98.25%	85.51%	100%	97%
Testing Accuracy	79.21%	84.04%	84.33%	87.68%
Training Kappa	0.97	0.71	1	0.94
Testing Kappa	0.58	0.68	0.69	0.75

Table 7-19: Summary of results for the Boston housing data set.

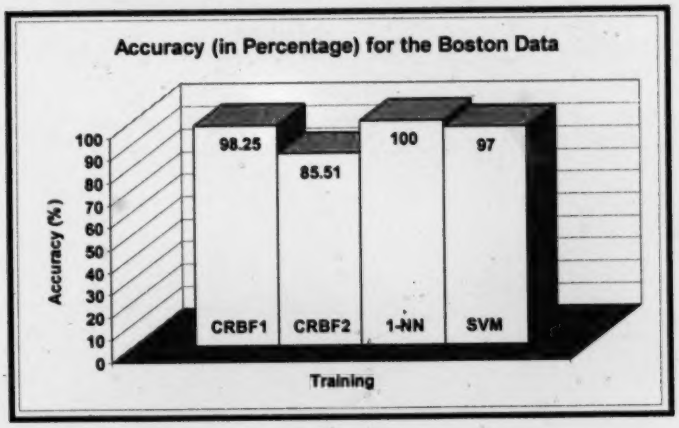


Figure 7-5: Training accuracy for the Boston housing data set.

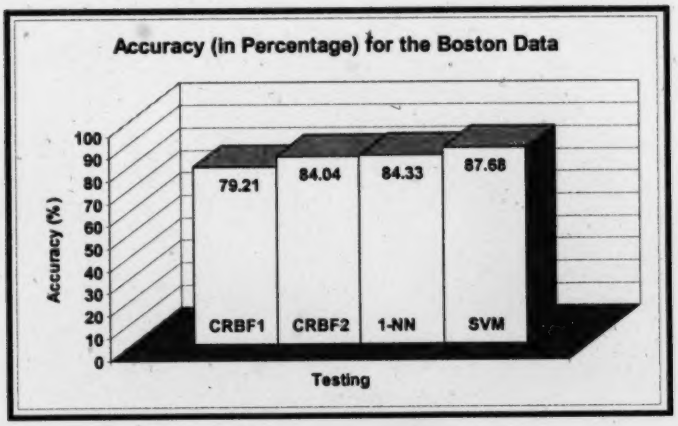


Figure 7-6: Testing accuracy for the Boston housing data set.

7.6 Storm Cell Classification

This study is concerned with data that were collected from the Vivian radar in Manitoba and processed by the RDSS. There are three groups of experiments: the first dealt with a two-class classification problem [63], [64], the second one with a multi-class classification problem for four classes [62], and the third one with a multi-class classification problem for ten classes [62].

7.6.1 Weather data (a): Hail vs. Tornado

In this first group of experiments, the data set was a subset of the original data. This data set consisted of 431 cells, of which 166 were confirmed to be hail patterns and 265 of which were labelled as tornado cells.

7.6.1.1 Methodology

In the experiments, 60% of the data set (i.e. 258 cells) was used as a training (learning) set and the remaining 40% (173 cells) was used as a testing set. To obtain reliable results, a rotation method was employed, repeating the experiments ten times. Subsequently, the training data for each one of the ten experiments were normalized to attain a zero mean

and one standard deviation. The values for the mean and the standard deviation were then used to normalize the testing data for each experiment.

7.6.1.2 Comparative analysis

The experiments were performed independently over the 10 randomly re-sampled groups. The results described in this section include:

1. The average and standard deviation (in parenthesis) confusion matrices for the training and testing sets.
2. The average kappa score (κ), including its standard deviation (in parenthesis), for the training and testing sets.
3. The average classification error (E), including its standard deviation (in parenthesis), for the training and testing sets.

The labels for the confusion matrices are:

- To: True (desired) output
- Co: Classifier (actual) output
- H: Hail cells
- T: Tornado cells

The performance results are displayed in Table 7-20 through Table 7-23.

Training		
To\Co	H	T
H	**98.2(6.1)**	2.1(1.1)
T	2.2(1.1)	**155.5(6.1)**

κ=0.97(0.01), E=1.67%(0.55%)

Testing		
To\Co	H	T
H	**42.8(5.8)**	22.9(5.4)
T	21.1(5.1)	**86.2(5.2)**

κ=0.46(0.07), E=25.43%(3.28%)

Table 7-20: CRBF1 training and testing performance results for the weather data (a).

Training		
To\Co	H	T
H	**21.7(11.3)**	78.6(8.2)
T	13.6(9.4)	**144.1(14.7)**

κ=0.14(0.07), E=35.74%(3.11%)

Testing		
To\Co	H	T
H	**13.2(6.6)**	52.5(11.3)
T	10.3(9.7)	**97(5.8)**

κ=0.13(0.06), E=36.30%(2.52%)

Table 7-21: CRBF2 training and testing performance results for the weather data (a).

Training		
To\Co	H	T
H	**100.3(6.0)**	0(0)
T	0(0)	**157.7(6.0)**

κ=1(0), E=0%(0%)

Testing		
To\Co	H	T
H	**47.8(4.3)**	17.9(4.0)
T	16.7(6.5)	**90.6(4.4)**

κ=0.57(0.07), E=20%(3.15%)

Table 7-22: 1-NN training and testing performance results for the weather data (a).

Training		
To\Co	H	T
H	**98.3(6.7)**	2(2.3)
T	1.1(1.9)	**156.6(6.3)**

κ=0.98(0.03), E=1.2%(1.51%)

Testing		
To\Co	H	T
H	**48.7(5.2)**	17(3.1)
T	14.8(4.2)	**92.5(5.6)**

κ=0.61(0.05), E=18.38%(2.43%)

Table 7-23: SVM training and testing performance results for the weather data (a).

The results averaged over the ten iterations are summarized in Table 7-24, Figure 7-7, and Figure 7-8. The results include the accuracy and the kappa score in training and testing. From these results, it can be seen that the SVM was the approach that showed the best accuracy (81.62% under testing) and the highest kappa score (0.61 under testing).

	CRBF1	CRBF2	1-NN	SVM
Training Accuracy	98.33%	64.26%	100%	98.8%
Testing Accuracy	74.57%	63.7%	80%	81.62%
Training Kappa	0.97	0.14	1	0.98
Testing Kappa	0.46	0.13	0.57	0.61

Table 7-24: Summary of results for the weather data set (a).

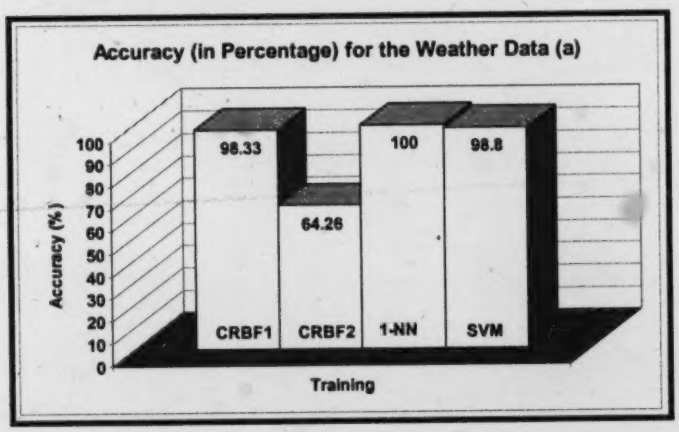


Figure 7-7: Training accuracy for the weather data (a). The classification problem is binary.

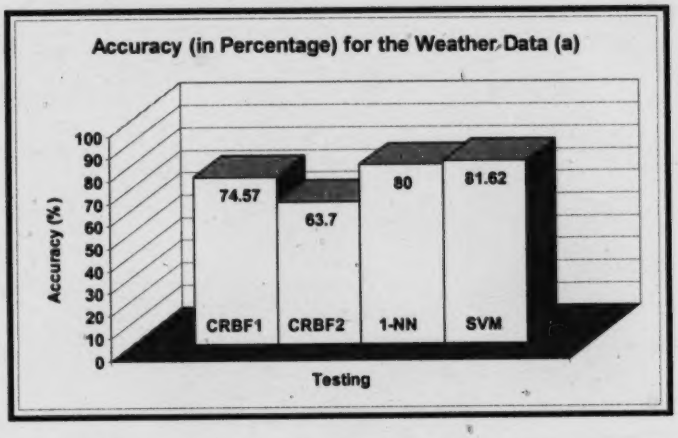


Figure 7-8: Testing accuracy for the weather data (a). The classification problem is binary.

7.6.1.3 Comparison with previous work

Table 7-25 shows the performance (accuracy, in percentage, on the test set) of the following methods for solving the storm cell classification task: a multilayer perceptron used by Alexiuk et al. [1], a linear discriminant analysis used by Li et al. [42], and the support vector machine-based classifier presented here. Direct comparison, however, could not be done because of the following reasons:

1. The experiments were conducted for different data sets (99 hail cells and 25 tornado cells in the case of Alexiuk et al. and 172 hail cells and 163 tornado cells in the case of Li et al.).
2. The pre-processing strategies were different (in the case of Alexiuk et al., they used PCA [39], fuzzy interquartile encoding, and genetic algorithms. In the case of Li et al., they used genetic algorithms for optimizing the derived products).
3. The experiments were conducted under different formats (Li et al. used leave one out to estimate the test error. Alexiuk et al. did not specify the percentages of training/testing data used for the experiments).

Multilayer Perceptron	Linear Discriminant Analysis	Support Vector Machines
78%	75.2%	81.62%

Table 7-25: Performance of different methods for solving a storm cell classification problem.

7.6.2 Weather data (b): Four-class Classification Problem

This data set consisted of 577 cells, of which 166 were confirmed to be hail patterns, 54 of which were labelled as rain cells, 265 of which were confirmed to be tornado cells, and 92 of which were labelled as wind cells.

7.6.2.1 Methodology

In the experiments, 60% of the data set (i.e. 346 cells) was used as a training (learning) set and the remaining 40% (231 cells) was used as a testing set. To obtain reliable results, a rotation method was employed, repeating the experiments ten times. Subsequently, the training data for each one of the ten experiments were normalized to attain a zero mean and one standard deviation. The values for the mean and the standard deviation were then used to normalize the testing data for each experiment.

7.6.2.2 Comparative analysis

The experiments were performed independently over the 10 randomly re-sampled groups. The results described in this section include:

1. The average and standard deviation (in parenthesis) confusion matrices for the training and testing sets.
2. The average kappa score (κ), including its standard deviation (in parenthesis), for the training and testing sets.
3. The average classification error (E), including its standard deviation (in parenthesis), for the training and testing sets.

The labels for the confusion matrices are:

- To: True (desired) output
- Co: Classifier (actual) output
- H: Hail
- R: Rain
- T: Tornado
- W: Wind

The performance results are displayed in Table 7-26 through Table 7-30.

Training				
To\Co	H	R	T	W
H	**95.7(6.3)**	2(1.8)	2.4(1.7)	0.1(0.3)
R	2.2(1.8)	**21.8(4.2)**	3.9(1.7)	4.2(1.7)
T	1.3(1.3)	1.5(1.1)	**153.1(2.9)**	2.6(2.1)
W	0.1(0.3)	1.5(1.5)	7.3(1.5)	**46.3(4.2)**

κ=0.87(0.02), E=8.41%(1.32%)

Testing				
To\Co	H	R	T	W
H	**35.4(7.0)**	8.4(4.1)	14.9(4.1)	7.1(3.1)
R	4.3(2.6)	**7.1(2.6)**	6.5(3.2)	4(1.5)
T	16.6(5.0)	10.9(3.9)	**67.5(6.8)**	11.5(4.6)
W	5.3(2.5)	4.8(2.5)	9.8(3.0)	**16.9(4.3)**

κ=0.34(0.07), E=45.07%(4.92%)

Table 7-26: CRBF1 training and testing set performance results for the weather data (b).

Training				
To\Co	H	R	T	W
H	**22.8(7.9)**	0(0)	77.3(6.8)	0.1(0.3)
R	0.2(0.6)	**0(0)**	31.9(4.2)	0(0)
T	13.7(5.3)	0(0)	**144.5(7.2)**	0.3(0.7)
W	6.7(4.1)	0(0)	48(5.3)	**0.5(1.3)**

κ=0.09(0.03), E=51.50%(1.59%)

Testing				
To\Co	H	R	T	W
H	**12(2.9)**	0(0)	53.8(7.1)	0(0)
R	0.1(0.3)	**0(0)**	21.8(4.6)	0(0)
T	9.2(4.3)	0.2(0.4)	**96(3.2)**	1.1(1.2)
W	3.8(2.4)	0(0)	32.8(3.9)	**0.2(0.6)**

κ=0.05(0.02), E=53.16%(1.11%)

Table 7-27: CRBF2 training and testing set performance results for the weather data (b).

Training				
To\Co	H	R	T	W
H	**100.2(4.7)**	0(0)	0(0)	0(0)
R	0(0)	**32.1(4.5)**	0(0)	0(0)
T	0(0)	0(0)	**158.5(2.9)**	0(0)
W	0(0)	0(0)	0(5.3)	**55.2(3.3)**

κ=1(0), E=0%(0%)

Testing				
To\Co	H	R	T	W
H	**39(3.8)**	5.9(2.4)	16.7(4.1)	4.2(1.7)
R	3.6(2.6)	**7.7(2.7)**	6.4(2.6)	4.2(2.0)
T	13.8(2.7)	8.7(3.2)	**70.8(4.9)**	13.2(3.4)
W	5.7(2.7)	5.2(2.5)	9.5(2.4)	**16.4(3.3)**

κ=0.38(0.03), E=42.04%(2.53%)

Table 7-28: 1-NN training and testing set performance results for the weather data (b).

Training				
To\Co	H	R	T	W
H	**96.9(5.9)**	1.3(1.2)	1.9(1.6)	0.1(0.3)
R	3.6(1.4)	**18(3.9)**	6(3.3)	4.5(2.0)
T	0.5(0.7)	0.6(0.7)	**155.8(3.2)**	1.6(1.1)
W	0.4(0.8)	1.6(1.9)	9.2(2.5)	**44(4.8)**

κ=0.86(0.03), E=9.05%(1.79%)

Testing				
To\Co	H	R	T	W
H	**41.3(3.6)**	5.2(2.8)	15.1(5.5)	4.2(2.6)
R	4.3(3.0)	**6(1.2)**	8.2(3.1)	3.4(1.2)
T	11.9(3.1)	5.9(2.9)	**81.1(7.0)**	7.6(4.0)
W	5.2(2.5)	3.9(2.9)	12.1(3.2)	**15.6(3.8)**

κ=0.43(0.04), E=37.66%(3.02%)

Table 7-29: DDAG training and testing set performance results for the weather data (b).

Training				
To\Co	H	R	T	W
H	**97(5.7)**	1.3(1.1)	1.9(1.2)	0(0)
R	3.1(1.5)	**18.2(3.9)**	5.9(2.4)	4.9(1.8)
T	0.6(0.8)	0.7(1.6)	**155.3(3.6)**	1.9(1.0)
W	0.4(0.7)	0.7(1.1)	9(2.6)	**45.1(5.0)**

κ=0.87(0.03), E=8.79%(1.65%)

Testing				
To\Co	H	R	T	W
H	**42.9(2.9)**	3.3(2.5)	14.9(5.0)	4.7(2.6)
R	4.4(3.0)	**6.6(1.6)**	7.7(2.9)	3.2(1.3)
T	13.3(3.1)	3.7(2.8)	**80.9(5.0)**	8.6(3.9)
W	6.1(2.9)	3(1.4)	11.4(3.0)	**16.3(4.0)**

κ=0.45(0.05), E=36.49%(3.09%)

Table 7-30: 1-v-R training and testing set performance results for the weather data (b).

The results averaged over ten iterations are summarized in Figure 7-9, Figure 7-10, and Table 7-31. The results include the accuracy in training and testing and the kappa score in training and testing. From these results, it can be seen that the SVM, in both multi-class approaches, outperformed the CRBF1, the CRBF2, and the 1-NN classifiers showing the best accuracy (63.51% under testing) and the highest kappa score (0.45 under testing). Between the two SVM approaches that were tested, the 1-v-R approach had better accuracy (63.51% against 62.34% under testing) and higher kappa score (0.45 against 0.43 under testing).

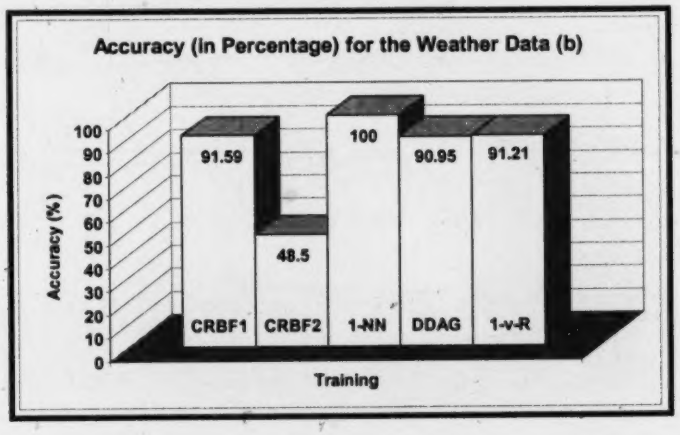


Figure 7-9: Training accuracy for the weather data (b). 4-class case.

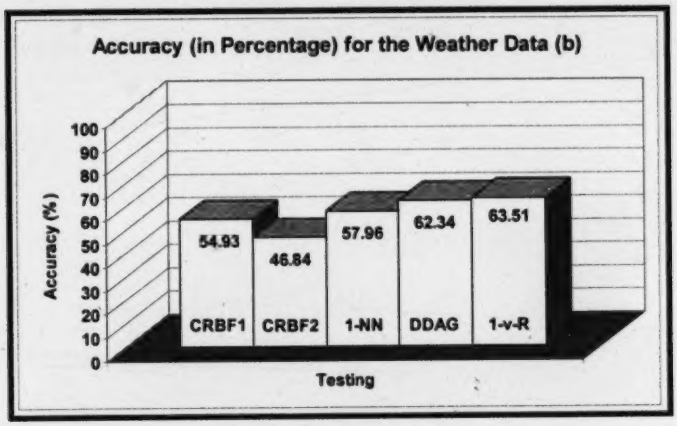


Figure 7-10: Testing accuracy for the weather data (b). 4-class case.

	CRBF1	CRBF2	1-NN	DDAG	1-v-R
Training Accuracy	91.59%	48.5%	100%	90.95%	91.21%
Testing Accuracy	54.93%	46.84%	57.96%	62.34%	63.51%
Training Kappa	0.87	0.09	1	0.86	0.87
Testing Kappa	0.34	0.05	0.38	0.43	0.45

Table 7-31: Summary of results for the weather data (b).

7.6.3 Weather data (c): Ten-class Classification Problem

This data set has ten classes that include the four original ones (hail, rain, tornado and wind) and six new classes that were added to deal with the patterns that had two different labels.

7.6.3.1 Methodology

In the experiments, 60% of the data set (i.e. 346 cells) was used as a training (learning) set and the remaining 40% (231 cells) was used as a testing set. To obtain reliable results, a rotation method was employed, repeating the experiments ten times. Subsequently, the training data for each one of the ten experiments were normalized to attain a zero mean and one standard deviation. The values for the mean and the standard deviation were then used to normalize the testing data for each experiment. The pattern distribution for each one of the ten experiments is depicted in Table 7-32.

Class	Name	Patterns	Training Patterns	Testing Patterns
1	Hail	150	112	38
2	Rain	22	16	6
3	Tornado	207	155	52
4	Wind	52	39	13
5	Hail or Rain	20	15	5
6	Hail or Tornado	10	7	3
7	Hail or Wind	0	0	0
8	Rain or Tornado	33	24	9
9	Rain or Wind	33	24	9
10	Tornado or Wind	50	37	13
	Totals	577	431	146

Table 7-32: Pattern distribution for the 10-class classification problem

7.6.3.2 Comparative analysis

The experiments were performed independently over the 10 randomly re-sampled groups. The results described in this section include:

1. The average and standard deviation confusion matrices for the training and testing sets.
2. The average kappa score (κ), including its standard deviation (in parenthesis), for the training and testing sets.
3. The average classification error (E), including its standard deviation (in parenthesis), for the training and testing sets.

The labels for the confusion matrices are:

- To: True (desired) output
- Co: Classifier (actual) output
- H: Hail
- R: Rain
- T: Tornado
- W: Wind
- HR: Hail or Rain
- HT: Hail or Tornado
- HW: Hail or Wind
- RT: Rain or Tornado
- RW: Rain or Wind
- TW: Tornado or Wind

The performance results are displayed in Table 7-33 through Table 7-52.

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**86.8**	0.0	1.3	0.0	0.0	0.0	0.0	0.0	0.2	0.0
R	0.0	**12.7**	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T	1.1	0.1	**124.4**	0.1	0.0	0.2	0.0	0.0	0.0	0.0
W	0.1	0.0	0.0	**30.7**	0.0	0.0	0.0	0.0	0.0	0.0
HR	0.1	0.0	0.0	0.0	**11.5**	0.0	0.0	0.0	0.0	0.0
HT	0.1	0.0	0.2	0.0	0.0	**5.5**	0.0	0.0	0.0	0.1
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**19.4**	0.0	0.0
RW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**21.0**	0.0
TW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**29.6**

κ=0.98(0.01), E=1.27%(0.74%)

Table 7-33: CRBF1 training set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**6.7**	0.0	1.6	0.0	0.0	0.0	0.0	0.0	0.4	0.0
R	0.0	**2.6**	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T	1.1	0.3	**5.2**	0.3	0.0	0.4	0.0	0.0	0.0	0.0
W	0.3	0.0	0.0	**3.1**	0.0	0.0	0.0	0.0	0.0	0.0
HR	0.3	0.0	0.0	0.0	**3.0**	0.0	0.0	0.0	0.0	0.0
HT	0.3	0.0	0.4	0.0	0.0	**2.1**	0.0	0.0	0.0	0.3
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**3.5**	0.0	0.0
RW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.6**	0.0
TW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**3.3**

Table 7-34: CRBF1 training set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**28.7**	2.7	14.7	4.6	1.1	1.3	0.0	0.7	4.0	3.9
R	0.2	**5.5**	1.6	0.4	0.0	0.0	0.0	0.4	0.0	0.4
T	10.5	1.9	**56.4**	3.7	0.3	1.6	0.0	1.1	2.9	2.7
W	2.4	0.1	3.1	**12.4**	0.3	0.4	0.0	0.2	1.1	1.2
HR	1.1	0.0	0.0	0.0	**7.3**	0.0	0.0	0.0	0.0	0.0
HT	0.6	0.0	0.8	0.0	0.0	**2.4**	0.0	0.0	0.0	0.3
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	1.2	0.0	0.6	0.0	0.0	0.0	0.0	**11.8**	0.0	0.0
RW	1.2	0.0	0.6	0.0	0.0	0.0	0.0	0.0	**10.2**	0.0
TW	1.0	0.3	1.2	0.6	0.0	0.2	0.0	0.0	0.0	**17.1**

κ=0.57(0.04), E=34.29%(3.17%)

Table 7-35: CRBF1 testing set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**4.7**	1.9	5.2	2.9	1.3	1.3	0.0	1.2	1.6	2.1
R	0.6	**2.3**	2.2	0.8	0.0	0.0	0.0	0.8	0.0	1.3
T	3.7	1.7	**3.5**	2.5	0.7	1.2	0.0	1.3	2.6	2.2
W	2.2	0.3	1.9	**2.6**	0.5	0.7	0.0	0.6	1.7	0.9
HR	1.8	0.0	0.0	0.0	**2.6**	0.0	0.0	0.0	0.0	0.0
HT	1.3	0.0	1.4	0.0	0.0	**1.0**	0.0	0.0	0.0	0.9
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	1.5	0.0	1.3	0.0	0.0	0.0	0.0	**2.6**	0.0	0.0
RW	2.1	0.0	1.3	0.0	0.0	0.0	0.0	0.0	**2.0**	0.0
TW	1.3	0.9	1.3	1.3	0.0	0.6	0.0	0.0	0.0	**3.1**

Table 7-36: CRBF1 testing set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**15.8**	0.0	72.2	0.0	0.0	0.0	0.0	0.3	0.0	0.0
R	0.5	**0.0**	12.6	0.0	0.0	0.0	0.0	0.0	0.1	0.3
T	10.4	0.0	**114.5**	0.0	0.0	0.0	0.0	0.8	0.0	0.2
W	4.3	0.0	26.5	**0.0**	0.0	0.0	0.0	0.0	0.0	0.0
HR	0.5	0.0	9.9	0.0	**0.0**	0.0	0.0	1.2	0.0	0.0
HT	0.0	0.0	5.9	0.0	0.0	**0.0**	0.0	0.0	0.0	0.0
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	1.6	0.0	16.3	0.0	0.0	0.0	0.0	**1.5**	0.0	0.0
RW	0.3	0.0	20.3	0.0	0.0	0.0	0.0	0.2	**0.0**	0.2
TW	0.1	0.0	28.6	0.0	0.0	0.0	0.0	0.3	0.1	**0.5**

κ=0.05(0.05), E=61.76%(2.24%)

Table 7-37: CRBF2 training set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**12.6**	0.0	11.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0
R	1.6	**0.0**	3.5	0.0	0.0	0.0	0.0	0.0	0.3	0.7
T	9.9	0.0	**11.2**	0.0	0.0	0.0	0.0	0.9	0.0	0.4
W	4.1	0.0	5.9	**0.0**	0.0	0.0	0.0	0.0	0.0	0.0
HR	1.1	0.0	3.7	0.0	**0.0**	0.0	0.0	2.6	0.0	0.0
HT	0.0	0.0	1.7	0.0	0.0	**0.0**	0.0	0.0	0.0	0.0
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	2.3	0.0	4.5	0.0	0.0	0.0	0.0	**4.1**	0.0	0.0
RW	0.9	0.0	2.3	0.0	0.0	0.0	0.0	0.6	**0.0**	0.6
TW	0.3	0.0	3.8	0.0	0.0	0.0	0.0	0.9	0.3	**0.8**

Table 7-38: CRBF2 training set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**9.3**	0.0	51.3	0.0	0.0	0.0	0.0	1.1	0.0	0.0
R	0.1	**0.0**	8.2	0.0	0.0	0.0	0.0	0.0	0.1	0.1
T	7.4	0.0	**73.0**	0.0	0.0	0.0	0.0	0.5	0.0	0.2
W	3.1	0.0	18.0	**0.0**	0.0	0.0	0.0	0.1	0.0	0.0
HR	0.3	0.0	7.7	0.0	**0.0**	0.0	0.0	0.4	0.0	0.0
HT	0.0	0.0	4.1	0.0	0.0	**0.0**	0.0	0.0	0.0	0.0
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	1.7	0.0	11.3	0.0	0.0	0.0	0.0	**0.6**	0.0	0.0
RW	0.3	0.0	11.5	0.0	0.0	0.0	0.0	0.1	**0.0**	0.1
TW	0.1	0.0	19.5	0.0	0.0	0.0	0.0	0.2	0.2	**0.4**

κ=0.04(0.03), E=63.94%(2.47%)

Table 7-39: CRBF2 testing set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**7.8**	0.0	12.9	0.0	0.0	0.0	0.0	2.1	0.0	0.0
R	0.3	**0.0**	3.0	0.0	0.0	0.0	0.0	0.0	0.3	0.3
T	8.5	0.0	**9.1**	0.0	0.0	0.0	0.0	1.3	0.0	0.6
W	2.4	0.0	2.4	**0.0**	0.0	0.0	0.0	0.3	0.0	0.0
HR	0.7	0.0	3.6	0.0	**0.0**	0.0	0.0	1.0	0.0	0.0
HT	0.0	0.0	1.7	0.0	0.0	**0.0**	0.0	0.0	0.0	0.0
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	3.7	0.0	5.6	0.0	0.0	0.0	0.0	**1.6**	0.0	0.0
RW	0.9	0.0	1.6	0.0	0.0	0.0	0.0	0.3	**0.0**	0.3
TW	0.3	0.0	3.4	0.0	0.0	0.0	0.0	0.6	0.6	**1.0**

Table 7-40: CRBF2 testing set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**88.3**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
R	0.0	**13.5**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T	0.0	0.0	**125.9**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
W	0.0	0.0	0.0	**30.8**	0.0	0.0	0.0	0.0	0.0	0.0
HR	0.0	0.0	0.0	0.0	**11.6**	0.0	0.0	0.0	0.0	0.0
HT	0.0	0.0	0.0	0.0	0.0	**5.9**	0.0	0.0	0.0	0.0
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**19.4**	0.0	0.0
RW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**21.0**	0.0
TW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**29.6**

κ=1(0), E=0%(0%)

Table 7-41: 1-NN training set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**5.8**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
R	0.0	**2.9**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T	0.0	0.0	**5.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
W	0.0	0.0	0.0	**3.1**	0.0	0.0	0.0	0.0	0.0	0.0
HR	0.0	0.0	0.0	0.0	**2.9**	0.0	0.0	0.0	0.0	0.0
HT	0.0	0.0	0.0	0.0	0.0	**1.7**	0.0	0.0	0.0	0.0
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**3.5**	0.0	0.0
RW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.6**	0.0
TW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**3.3**

Table 7-42: 1-NN training set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**39.6**	1.2	13.2	4.0	0.3	0.1	0.0	1.3	1.3	0.7
R	0.2	**5.5**	2.0	0.2	0.0	0.0	0.0	0.0	0.6	0.0
T	8.3	1.8	**60.4**	2.5	0.0	0.8	0.0	1.3	3.0	3.0
W	3.2	0.0	2.7	**13.1**	0.4	0.2	0.0	0.0	1.0	0.6
HR	0.4	0.0	0.0	0.0	**7.6**	0.0	0.0	0.4	0.0	0.0
HT	0.0	0.0	0.7	0.0	0.0	**3.4**	0.0	0.0	0.0	0.0
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.9	0.0	0.3	0.0	0.0	0.0	0.0	**12.4**	0.0	0.0
RW	0.6	0.6	0.0	0.0	0.0	0.0	0.0	0.0	**10.5**	0.3
TW	0.8	0.0	0.3	0.4	0.0	0.2	0.0	0.5	0.2	**18.0**

κ=0.67(0.04), E=26.19%(3.07%)

Table 7-43: 1-NN testing set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**4.7**	0.8	4.8	2.5	0.5	0.3	0.0	1.1	1.2	0.9
R	0.6	**1.9**	2.1	0.6	0.0	0.0	0.0	0.0	1.0	0.0
T	4.3	1.3	**5.3**	2.2	0.0	1.0	0.0	1.1	1.4	2.1
W	1.9	0.0	1.2	**2.7**	0.5	0.4	0.0	0.0	1.4	0.8
HR	1.3	0.0	0.0	0.0	**2.6**	0.0	0.0	1.3	0.0	0.0
HT	0.0	0.0	1.5	0.0	0.0	**1.4**	0.0	0.0	0.0	0.0
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	1.4	0.0	0.9	0.0	0.0	0.0	0.0	**2.7**	0.0	0.0
RW	1.3	1.3	0.0	0.0	0.0	0.0	0.0	0.0	**1.7**	0.9
TW	1.3	0.0	0.9	0.8	0.0	0.6	0.0	1.1	0.6	**3.7**

Table 7-44: 1-NN testing set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**84.6**	0.0	2.3	0.5	0.1	0.0	0.0	0.5	0.3	0.0
R	0.0	**8.9**	3.1	0.2	0.0	0.0	0.0	1.3	0.0	0.0
T	1.5	0.1	**122.5**	0.5	0.2	0.2	0.0	0.4	0.5	0.0
W	0.4	0.2	0.7	**28.9**	0.1	0.2	0.0	0.0	0.0	0.3
HR	1.0	0.0	0.6	0.0	**9.2**	0.0	0.0	0.6	0.0	0.2
HT	0.6	0.0	2.5	0.6	0.2	**1.7**	0.0	0.0	0.0	0.3
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.0	0.0	0.7	0.1	0.2	0.0	0.0	**18.3**	0.0	0.1
RW	0.2	0.1	0.5	0.1	0.0	0.0	0.0	0.2	**19.9**	0.0
TW	0.0	0.0	0.2	0.3	0.1	0.2	0.0	0.4	0.5	**27.9**

κ=0.91(0.03), E=6.97%(2.54%)

Table 7-45: DDAG training set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**7.7**	0.0	3.4	0.7	0.3	0.0	0.0	0.7	0.7	0.0
R	0.0	**4.3**	2.4	0.4	0.0	0.0	0.0	1.2	0.0	0.0
T	2.5	0.3	**5.1**	0.7	0.4	0.4	0.0	0.7	0.8	0.0
W	0.7	0.4	1.5	**3.1**	0.3	0.4	0.0	0.0	0.0	0.5
HR	2.2	0.0	0.8	0.0	**3.9**	0.0	0.0	1.6	0.0	0.6
HT	1.3	0.0	1.6	1.3	0.6	**2.9**	0.0	0.0	0.0	0.9
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.0	0.0	1.6	0.3	0.4	0.0	0.0	**4.1**	0.0	0.3
RW	0.6	0.3	0.7	0.3	0.0	0.0	0.0	0.4	**2.0**	0.0
TW	0.0	0.0	0.4	0.7	0.3	0.4	0.0	0.7	0.5	**3.3**

Table 7-46: DDAG training set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**38.4**	2.0	12.6	2.9	1.4	0.3	0.0	1.2	2.1	0.8
R	0.4	**2.7**	3.1	0.6	0.4	0.0	0.0	1.3	0.0	0.0
T	7.2	1.0	**63.9**	2.6	0.3	1.0	0.0	2.2	0.6	2.3
W	2.4	0.2	3.0	**14.5**	0.2	0.1	0.0	0.0	0.2	0.6
HR	1.4	0.0	1.0	0.0	**4.8**	0.0	0.0	0.6	0.0	0.6
HT	0.3	0.3	2.8	0.3	0.1	**0.2**	0.0	0.0	0.0	0.1
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.6	0.0	0.8	0.5	0.7	0.0	0.0	**10.8**	0.0	0.2
RW	1.0	0.5	1.3	0.2	0.0	0.3	0.0	0.4	**8.3**	0.0
TW	0.4	0.0	1.7	0.7	0.2	0.4	0.0	0.7	1.4	**14.9**

κ=0.60(0.04), E=31.39%(3.04%)

Table 7-47: DDAG testing set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**4.2**	1.9	3.7	2.6	1.2	0.7	0.0	0.8	1.3	0.6
R	0.8	**0.9**	2.9	0.8	0.8	0.0	0.0	0.7	0.0	0.0
T	3.3	1.2	**3.5**	1.9	0.5	1.1	0.0	1.3	0.8	1.1
W	1.8	0.4	2.2	**3.3**	0.4	0.3	0.0	0.0	0.6	0.7
HR	1.9	0.0	1.3	0.0	**2.4**	0.0	0.0	1.3	0.0	1.9
HT	0.7	0.9	1.9	0.7	0.3	**0.4**	0.0	0.0	0.0	0.3
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	1.3	0.0	1.5	1.1	1.6	0.0	0.0	**2.7**	0.0	0.6
RW	2.2	1.6	1.5	0.6	0.0	0.9	0.0	0.8	**2.1**	0.0
TW	0.8	0.0	1.6	1.1	0.6	0.8	0.0	1.1	1.8	**3.3**

Table 7-48: DDAG testing set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**88.3**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
R	0.1	**13.0**	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T	0.0	0.0	**125.9**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
W	0.1	0.0	0.0	**30.7**	0.0	0.0	0.0	0.0	0.0	0.0
HR	0.0	0.0	0.0	0.0	**11.6**	0.0	0.0	0.0	0.0	0.0
HT	0.0	0.0	0.1	0.0	0.0	**5.7**	0.0	0.0	0.0	0.1
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.2	0.0	0.0	0.0	0.0	0.0	0.0	**19.1**	0.0	0.1
RW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**21.0**	0.0
TW	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	**29.5**

κ=1.00(0.01), E=0.35%(0.47%)

Table 7-49: 1-v-R training set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**5.8**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
R	0.3	**3.3**	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T	0.0	0.0	**5.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
W	0.3	0.0	0.0	**3.3**	0.0	0.0	0.0	0.0	0.0	0.0
HR	0.0	0.0	0.0	0.0	**2.9**	0.0	0.0	0.0	0.0	0.0
HT	0.0	0.0	0.3	0.0	0.0	**1.9**	0.0	0.0	0.0	0.3
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.6	0.0	0.0	0.0	0.0	0.0	0.0	**4.0**	0.0	0.3
RW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.6**	0.0
TW	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	**3.5**

Table 7-50: 1-v-R training set performance results (standard deviation) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**42.6**	1.0	10.0	3.1	1.0	0.1	0.0	0.8	0.9	2.2
R	0.7	**5.4**	1.4	0.2	0.0	0.0	0.0	0.4	0.4	0.0
T	8.2	1.0	**63.7**	2.1	0.4	0.1	0.0	1.5	0.8	3.3
W	2.0	0.1	2.8	**14.9**	0.3	0.1	0.0	0.0	0.0	1.0
HR	0.4	0.0	0.0	0.0	**7.6**	0.0	0.0	0.0	0.0	0.4
HT	0.0	0.0	1.0	0.0	0.0	**2.8**	0.0	0.0	0.0	0.3
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	0.7	0.0	0.3	0.0	0.0	0.0	0.0	**12.4**	0.0	0.2
RW	0.9	0.0	0.3	0.0	0.0	0.0	0.0	0.0	**10.8**	0.0
TW	0.8	0.0	0.2	0.8	0.0	0.1	0.0	0.0	0.0	**18.5**

κ=0.71(0.04), E=22.64%(2.78%)

Table 7-51: 1-v-R testing set performance results (average) for the weather data (c).

To\Co	H	R	T	W	HR	HT	HW	RT	RW	TW
H	**6.3**	0.9	4.8	2.2	1.1	0.3	0.0	0.9	0.6	1.5
R	0.9	**0.8**	1.6	0.6	0.0	0.0	0.0	0.8	0.8	0.0
T	2.7	1.2	**3.3**	1.4	0.5	0.3	0.0	1.4	1.1	1.6
W	1.8	0.3	1.9	**3.5**	0.7	0.3	0.0	0.0	0.0	1.1
HR	1.3	0.0	0.0	0.0	**2.6**	0.0	0.0	0.0	0.0	1.3
HT	0.0	0.0	1.6	0.0	0.0	**0.9**	0.0	0.0	0.0	0.9
HW	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
RT	1.3	0.0	0.9	0.0	0.0	0.0	0.0	**2.7**	0.0	0.6
RW	1.4	0.0	0.9	0.0	0.0	0.0	0.0	0.0	**1.7**	0.0
TW	1.3	0.0	0.6	1.8	0.0	0.3	0.0	0.0	0.0	**3.5**

Table 7-52: 1-v-R testing set performance results (standard deviation) for the weather data (c).

The results averaged over ten iterations, for the 10-class case, are summarized in Table 7-53, Figure 7-11, and Figure 7-12. The results include the accuracy in training and testing and the kappa score in training and testing. From these results, it can be seen that the SVM, in the 1-v-R approach, outperformed all the other classifiers showing the best accuracy (77.36% under testing) and the highest kappa score (0.71 under testing). From these results, it can also be seen that the CRBF2 classifier performed poorly in this 10-class classification problem. This poor performance may be caused by the small number of receptive fields the CRBF2 is using in solving this problem. In fact, while the CRBF1 used an average of 199 receptive fields, the CRBF2 used an average of 11 receptive fields. This small number of receptive fields is associated with a small number of clusters (determined by prototypes stability analysis) that may not adequately represent the structure in the data. Therefore, experiments involving a larger maximum number of clusters to be used in the prototypes

stability analysis are required to properly study the performance of the CRBF2 classifier in solving this 10-class classification problem.

	CRBF1	CRBF2	1-NN	DDAG	1-v-R
Training Accuracy	98.73%	38.24%	100%	93.03%	99.65%
Testing Accuracy	65.71%	36.06%	73.81%	68.61%	77.36%
Training Kappa	0.98	0.05	1	0.91	1.0
Testing Kappa	0.57	0.04	0.67	0.60	0.71

Table 7-53: Summary of results for the weather data (c).

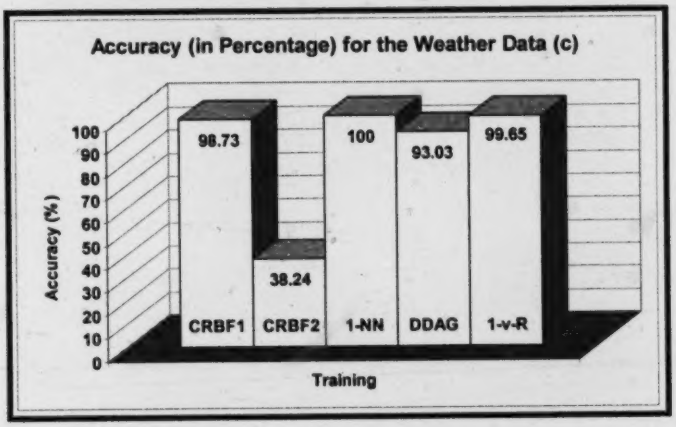


Figure 7-11: Training accuracy for the weather data (c). 10-class case.

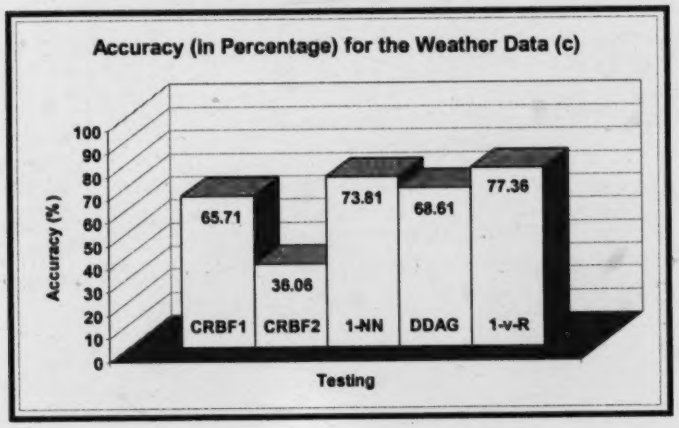


Figure 7-12: Testing accuracy for the weather data (c). 10-class case.

7.7 Discussion

Through the experiments presented in this chapter, it has been found that the SVM approach generally outperforms other classification methods in terms of higher classification accuracy. The good results obtained with the SVM may be related to the fact that the SVM produces decision surfaces that maximize the margin, as illustrated in Section 4.3. This fact makes it more robust than other classifiers in dealing with imprecise class labels.

In this chapter, the notion of prototypes stability analysis (whose intent is to determine the appropriate number of clusters, prototypes, or information granules to be used in classifier design) was applied to determine the number of centres for the design of the receptive fields of an RBF neural network (CRBF2). Even though the classification rates were not the best ones, the results in training and testing were quite similar (a difference of less than 3%). This may indicate that certain structure in the data was found and that most work must be done in the recognition stage to improve the classification rates.

Finally, it could be observed that the results were variable for the accuracy of the different methods. The greatest variability was associated with the weather data set. This may be related to the fact that this particular data set has a large number of data patterns that are identical but have different labels (see Section 2.6.5). Therefore, only the classifiers that were robust in dealing with imprecise class labels produced similar results.

7.8 Summary

This chapter compared the performance of the methods described in the previous sections (i.e. the RBF and the SVM approaches) with the performance of 1-nearest neighbour classifier. These studies relied on widely available data sets, an iris data set, a Wisconsin breast cancer data set and a Boston housing data, and a real-life data set, a severe storm cell data set from Environment Canada.

It has been found through these experiments that, when compared to classic radial basis function neural networks and the 1-nearest neighbour classifier, the support vector machine approach achieved superior classification accuracy in most of the classification tasks. The good results obtained with the SVM may be related to the fact that the SVM generally produces smooth decision surfaces that maximize the margin as depicted in the artificial data classification problem of Section 4.3.

Finally, the applicability of the prototypes stability analysis for determining the number of prototypes (in the case of the CRBF2) to be used in classifier design was also studied. The results suggest that this technique could be applied with success in future applications.

8 Conclusions and Recommendations for Future Work

8.1 Conclusions

This thesis had two main objectives. The first objective was to study the potential of support vector machines and radial basis function neural networks when applied to a series of classification problems. The second objective was to enhance these two approaches by using ideas of granular computing.

Regarding the first objective, this thesis has assessed the applicability of the support vector machine for a series of classification tasks that include iris classification, breast cancer classification, housing price classification, and storm cell classification. Compared to classic radial basis function neural networks and 1-NN classifiers the support vector machine achieved superior classification accuracy in all the cases but in the breast cancer classification in which its performance was similar to that of the other classifiers. The decision directed acyclic graph and one-versus-rest strategies were used with the support vector machine in order to deal with the above multi-class (3 in the case of the iris data set and 4 and 10 in the case of the storm cell data) problems. The latter strategy consistently outperformed the former with respect to accuracy. The good results obtained with the SVM may be related to the fact that the SVM generally produces smooth decision surfaces that maximize the margin as depicted in the artificial data classification problem of previous chapters. This fact makes it more robust than the other classifiers in dealing with imprecise class labels.

Regarding the second objective, this thesis presented an attempt to incorporate fuzzy methods into the support vector machine (SVM) approach. This may be the first step to develop an SVM approach capable of using previous knowledge, encoded in the form of fuzzy rules, during the learning process and capable of producing interpretable results by transforming the parameters learned back to fuzzy rules. This SVM approach was called hybrid support vector machine (HSVM) because some computations are performed using operations on fuzzy sets. These operations consisted of similarity measures realized through a special type of kernels that were called fuzzy kernels because they operate on fuzzy sets. These fuzzy kernels were optimized using genetic algorithms. The HSVM performance was studied using artificial classification problems. The initial results obtained suggest that even though more work must be done to make sure the HSVM maximizes the margin, the accuracy in testing is promising enough to justify further research.

One of the most important considerations in classifier design is the selection of the appropriate number of clusters, prototypes, or information granules. In this thesis, a concept of prototypes stability was introduced. Prototypes are said to be stable if the variation among dimensions of the resulting prototypes corresponding to the same cluster in different subsets of the same data is minimal. Based on this idea, prototypes stability analysis (PSA) was used to determine an appropriate number of prototypes, clusters, or information granules to be used in classifier design. In this thesis, PSA was used to determine the number of centres for the design of the receptive fields of a radial basis function (RBF) neural network. Fuzzy c-means was used to find the centres of the receptive fields. From the results obtained, it could be seen that even though the classification rates were not always the best ones, the results in training and testing were quite similar indicating that the prototype stability analysis helped to find the best possible structure for classifier design.

8.2 Future Work

Below is a list of some of areas that should be considered for further work.

Hybrid support vector machines:

Among the aspects that could be considered for future investigation are the following:

1. Develop alternative methods for information granules generation.
2. Perform new experiments involving a fitness function that also takes into account the margin to produce decision surfaces that maximize the margin leading to better generalization performances.
3. Develop new kernels to generate different types of decision surfaces.
4. Use alternative optimization algorithms, such as gradient descent, to optimize the parameters associated with the kernels. These methods should focus in the reduction of the computational load associated with the use of genetic algorithms in the present approach.
5. Develop a methodology for interpreting the results obtained.

Prototypes stability analysis:

The future investigations should be centred in applying the proposed method in the design of other type of classifiers and in the development of rule-based systems.

Storm cell classification:

The main avenue for future investigations would be to evaluate complementary pre-processing strategies involving fuzzy set theory [77], [50], [54], [57]. Doing so, it will be possible to deal with imprecise class labels.

References

[1] M. Alexiuk, N, Pizzi, and W. Pedrycz, "Classification of Volumetric Storm Cell Patterns," in *Canadian Conference on Electrical and Computer Engineering*, Edmonton, 1999.

[2] E. Anderson, "The Irises of the Gaspe Peninsula," *Bulletin of the American Iris Society*, vol. 59, pp.2-5, 1935.

[3] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

[4] O. Barzilay and V. L. Brailovsky, "On Domain Knowledge and Feature Selection using Support Vector Machine," *Pattern Recognition Letters*, vol. 20, pp. 475-484, 1999.

[5] D. Belsley, E. Kuh, and R. Welsh, *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. New York: Wiley, 1980.

[6] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Functions*. New York: Plenum Press; 1981.

[7] J. C. Bezdek, "On the relationship between neural networks, pattern recognition and intelligence," *International Journal of Approximate Reasoning*, vol. 6, pp. 85-107, 1992.

[8] J. C. Bezdek, J. M. Keller, R. Krishnapuram, L. I. Kuncheva, and N. R. Pal, "Will the Real Iris data stand up?" *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 3, pp. 368-369, 1999.

[9] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Claredon Press, 1995.

[10] L. Bobrowski and J. C. Bezdek, "C-Means Clustering with the l_1 and l_∞ Norms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 545-554, 1991.

[11] P. P. Bonissone, "Soft Computing the Convergence of Emerging Reasoning Technologies," *Soft Computing*, vol.1, no. 1, pp. 6-18, 1997.

[12] B. E. Boser, I. M. Guyon and V. N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. D. Haussler ed., pp. 144-152, Pittsburg PA: ACM Press, 1992.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA.: Wadsworth International Group, 1984.

[14] C. J. C. Burges, "A tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[15] C. Campbell, "Algorithmic Approaches to Training Support Vector Machines: A Survey," in *Proceedings of ESANN 2000*, pp. 27-36. Belgium: D-Facto Publications, 2000.

[16] A. Carpenter and S. Grossberg, "A Massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision Graphics and Image Processing*, no. 37, pp. 54-115, 1983.

[17] A. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog output patterns," *Applied Optics*, vol.26, no. 23, pp. 4919-4930, 1987.

[18] A. Carpenter and S. Grossberg, "ART 3 Hierarchical Search: Chemical Transmitter in Self-organizing Pattern Recognition Architectures," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN'90)*, pp. 30-33. Washington, DC, 1990.

[19] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, vol. 2, no 2, pp. 302-309, 1991.

[20] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, pp. 273-297, 1995.

[21] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21-27, 1967.

[22] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge: University Press, 2000.

[23] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons, 1973.

[24] D. Dumitrescu, B. Lazzerini, and L. C. Jain, *Fuzzy Sets and their Application to Clustering and training*, New York: CRC Press, 2000.

[25] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter Control in Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124-141, 1999.

[26] J. L. Fleiss, "Measuring Agreement Between Two Judges on the Presence or Absence of a Trait," *Biometrics*, vol. 31, pp. 651-659, 1975.

[27] D. B. Fogel, "Review of Computational Intelligence: Imitating Life," *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1562-1565, 1995.

[28] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Second ed., Toronto: Academic Press, 1990.

[29] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Don Mills, Ontario: Addison-Wesley, 1989.

[30] D. E. Golberg and K. Deb, "A Comparative Analysis of Selection Schemes used in GA's," in *Foundation of Genetic Algorithms*, pp. 69-73. G. J. E. Rawlin. Ed. San Mateo California: Morgan Kaufmann, 1991.

[31] D. Harrison and D. L. Rubinfeld, "Hedonic Prices and the Demand for Clean Air," *Journal of Environmental Economics and Management*, vol. 5, pp. 81-102, 1978.

[32] F. Herrera and M. Lozano, "Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers," in *Genetic Algorithms and Soft Computing*. F. Herrera and J. L. Verdegay, eds., Heidelberg: Physica-Verlag, 1996.

[33] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review*, vol. 12, pp. 265-319, 1998.

[34] K. Hirota and W. Pedrycz, "Fuzzy Computing for Data Mining," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1575-1600, 1999.

[35] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Cambridge: MIT Press, 1975.

[36] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, N. J.: Prentice Hall, 1988.

[37] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial Neural Networks: A Tutorial," *Computer*, pp. 31-44, March 1996.

[38] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no 1, pp. 4-37, 2000.

[39] I. T. Jollife, *Principal Component Analysis*. New York: Springer-Verlag, 1986.

[40] T. Kohonen. *Self-Organizing Maps*. New York: Springer-Verlag, 1995.

[41] B. Kosko, *Neural Networks for Signal Processing*. New Jersey: Prentice Hall, 1992.

[42] P. C. Li, N. Pizzi, W. Pedrycz, D. Westmore, and R. Vivanco, "Severe Storm Cell Classification Using Derived Products Optimized by Genetic Algorithm," in *Canadian Conference on Electrical and Computer Engineering*, Halifax, 2000.

[43] R. P. Lippmann, "An introduction to Computing with Neural Nets," in *Fuzzy Models for Pattern recognition: Methods that Search for Structures in Data*. J. C. Bezdek and S. K. Pal, eds., IEEE CS Press, 1992.

[44] J. Lis and M. Lis, "Self-Adapting Parallel Genetic Algorithm with the Dynamic Mutation Probability, Crossover Rate and Population Size," in *Proceedings of the First Polish National Conference on Evolutionary computation*. J. Arabas, Ed. Oficina Wydawnica Politechniki Warszawskiej, 1996, pp. 324-329.

[45] O. L. Mangasarian and W. H. Woldberg, "Cancer Diagnosis Via Linear Programming," *SIAM News*, vol. 23, no. 5, pp. 1-18, 1990.

[46] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge: MIT Press, 1996.

[47] E. Osuna, R. Freund, and F. Girosi, "Support Vector Machines: Training and Applications," *MIT Artificial Intelligence Memo 1602*; MIT A.I. Lab, March 1997.

[48] E. Osuna, R. Freund, and F. Girosi, "Improved Training Algorithms for Support Vector Machines," in *Proceedings of IEEE Conference on Neural Networks and Signal Processing*; 1997.

[49] N. R. Pal and J. C. Bezdek, "On Cluster Validity for the Fuzzy c-Means Model," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 370-379, 1995.

[50] W. Pedrycz, "Fuzzy Sets in Pattern Recognition: Methodology and methods," *Pattern Recognition*, vol. 23, no.1 / 2, pp. 121-146, 1990.

[51] W. Pedrycz, "Direct and Inverse Problem in Comparison of Fuzzy Data," *Fuzzy Sets and Systems*, vol. 34, pp. 223-235, 1990.

[52] W. Pedrycz, "Neurocomputations in Relational Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 289-296, 1991.

[53] W. Pedrycz, "Fuzzy Neural Networks and Neurocomputation," *Fuzzy Sets and Systems*, vol. 56, no.1, pp. 1-28, 1993.

[54] W. Pedrycz, "Fuzzy Sets in Pattern Recognition: Accomplishments and Challenges," *Fuzzy Sets and Systems*, vol. 90, no. 2, pp. 171-176, 1997.

[55] W. Pedrycz and F. Gomide, *An Introduction to Fuzzy Sets: Analysis and Design*. Cambridge: MIT Press, 1998.

[56] W. Pedrycz and A. Vasilakos, "Linguistic Models and Linguistic Modeling," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 29, no. 6, pp. 745–757, 1999.

[57] N. J. Pizzi, "Fuzzy Pre-processing of Gold Standards as Applied to Biomedical Spectra Classification," *Artificial Intelligence in Medicine*, vol. 16, pp. 171-182, 1999.

[58] J. Platt, "Fast Training of Support Vector Machines using sequential Minimal Optimization," in *Advances in Kernel Methods: Support Vector Learning*. B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds., Cambridge, Mass.: MIT Press, 1999.

[59] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. "Large Margin DAGs for Multiclass Classification," in *Advances in Neural Information Processing Systems*, vol. 12, pp. 547-553. S. A Solla, T. K. Leen, and K. R. Müller, eds., Cambridge: MIT Press, 2000.

[60] T. Poggio and F. Girosi, "Networks for Approximation and Learning," in *Proceedings of the IEEE*, vol. 78, pp. 1481-1497, 1990.

[61] M. Pontil and A. Verri, "Properties of Support Vector Machines," *Neural Computation (MIT AI memo 1612)*, pp.1-17, 1998.

[62] L. Ramirez, W. Pedrycz, and N. Pizzi, "Storm cell Classification with the Use of Support Vector Machines," *Pattern Recognition*, submitted.

[63] L. Ramirez, W. Pedrycz, and N. Pizzi, "Severe Storm Cell Classification Using Support Vector Machines and Radial Basis Function Approaches," in *Canadian Conference on Electrical and Computer Engineering*, Toronto, 2001.

[64] L. Ramirez, W. Pedrycz, and N. Pizzi, "Classification of Severe Storm Cells Using Support Vector Machines," *in 6th Online World Conference on Soft Computing in Industrial Applications*, online, 2001.

[65] L. Ramirez and W. Pedrycz, "Prototypes Stability Analysis in the Design of Radial Basis Function Neural Networks," *Internal Report*, University of Alberta, 2001.

[66] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, pp. 533-536, 1986.

[67] R. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons Inc.; New York, 1992.

[68] B. Schölkopf. "Support Vector Learning". *PhD Dissertation*. Munich: R. Oldenbourg Verlag; 1997.

[69] B. Schölkopf, C. Burges, and V. Vapnik, "Extracting Support Data for a Given Task," in *Proceedings First International Conference on Knowledge Discovery and Data Mining*, pp. 252-257, 1995.

[70] C. Stone, M. Hansen, C. Kooperberg, and Y. Throung, "Polynomial Splines and their Tensor Products in Extended Linear Modeling (with discussion)," *Annals of Statistics*, vol. 25, no. 4, pp. 1371-1470, 1997.

[71] V. Vapnik, *Statistical Learning Theory*, New York: John Wiley & Sons, 1998.

[72] V. N. Vapnik, "An Overview of Statistical Learning Theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988-999, 1999.

[73] S. Watanabe, *Pattern Recognition: Human and Mechanical*, New York: Wiley, 1995.

[74] D. Westmore, "Radar Decision Support System (RDSS): High Level Design," *Technical Document*, InfoMagnetics Technologies Corporation 1997.

[75] J. Weston and C. Watkins, "Support Vector Machines for multi-class pattern recognition," in *Proceedings of the 6th European Symposium on Artificial Neural Networks (ESANN)*, 1999.

[76] A. H. Wright, "Genetic Algorithms for Real Parameter Optimization," in *Foundations of Genetic Algorithms 1*. G. J. E. Rawlin, ed., San Mateo: Morgan Kaufmann, 1991.

[77] L. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Process," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, pp. 28-44, 1973.

[78] L. Zadeh, "Fuzzy Logic, Neural Networks, and Soft Computing," *Communications of the ACM*, vol. 37, no. 3, pp. 77-84, 1994.

[79] L. Zadeh, "Toward a Theory of Fuzzy Information Granulation and its Centrality in Human Reasoning and Fuzzy Logic," *Fuzzy Sets and Systems*, vol. 90, pp. 111-127, 1997.

[80] L. Zadeh, "From Computing with Numbers to Computing with Words-From Manipulation of Measurements to Manipulation of Perceptions," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 45, no. 1, pp. 105-119, 1999.